Efektywne sieci komparatorów

Marcin Kik

praca doktorska

promotor: Prof. dr hab. Mirosław Kutyłowski

Instytut Informatyki
Uniwersytet Wrocławski

Wrocław, 2000

Efficient comparator networks

Marcin Kik

Ph.D. Thesis

supervisor: Prof. dr hab. Mirosław Kutyłowski

Institute of Computer Science
University of Wrocław

Wrocław, 2000

Acknowledgments

I would like to thank for an excellent cooperation and many helpful and inspiring discussions to Mirosław Kutyłowski, Krzysztof Loryś, Marek Piotrów and Grzegorz Stachowiak.

Our investigations have been initiated as a response to questions presented by Friedhelm Meyer auf der Heide, whom we owe gratitude for showing an exciting research area.

I would like also to thank Mirosław Kutyłowski and Krzysztof Loryś for their guidance, hints regarding formulation of the proofs, and last but not least for checking a lot of intermediate constructions that I have proposed.

Many thanks to Prof. Uwe Schwiegelshohn for pointing out many technical mistakes in the first version of this thesis.

Contents

1 Introduction

Sorting is one of the fundamental problems in data processing. Many operations can be performed much more efficiently on sorted data. There are many sorting algorithms. Majority of them are programs for RAM machine (i.e. for a classical model of computer). Many of the sorting algorithms have been invented for parallel (multiprocessor) computers with specific models of inter-processor connections. One of the other approaches is to invent a specialized hardware for sorting and related problems. A very popular approach in this area are *comparator networks*.

1.1 Comparator networks

A *comparator* is a simple device with two inputs and two outputs. For two numbers x and y arriving on the first and the second input, in a single computation step the comparator outputs the value $\min\{x, y\}$ on the first output and the value $\max\{x, y\}$ on the second output (see Fig. 1). Thus a comparator sorts a sequence of length two. We may pipeline two comparators so that an output of the first comparator will be used as an input of the second one. The second comparator can perform its computation, once the first comparator is finished.

A set of comparators with connections described is called a *comparator network*. There is a restriction that no loop-backs are allowed (i.e. no comparator network may contain a sequence of comparators $c_0, \ldots, c_k$, such that for all i, an output of $c_{i \bmod (k+1)}$ is connected to an input of $c_{(i+1) \bmod (k+1)}$). The input of the network is placed on the unconnected inputs of the comparators (called *inputs of the network*) and the output is taken from the unconnected outputs (called *outputs of the network*). The *input size* of a network is the number of its inputs. Since each comparator has two inputs and two outputs, the number of inputs of any network is equal to the number of its outputs.

If the input size of the network N is n, then we label the comparator inputs with n distinct integers $R_1, \ldots, R_n$ as follows:

- Each input of the network is labeled by one of the numbers $R_1, \ldots, R_n$, each number used for only one input.
- For each comparator c that has the first input labeled R_i and the second input labeled R_j, for some $i \neq j$, the first output of c is also labeled by R_i and the second one by R_j (as on Fig. 1).
- If an input of a comparator is connected to an output labeled R_i, then it is also labeled R_i.

For any network N with the input labeled by $R_1, \ldots, R_n$, for any ordered set X, the *input configuration over* X is a function $c : \{R_1, \ldots, R_n\} \to X$ such that the value $c(R_i)$ is placed on the network input labeled R_i. Let $c'(R_i)$ denote the value computed by N on the output labeled R_i for such an input. Then the function $c' : \{R_1, \ldots, R_n\} \to X$ is the *output* configuration for the input configuration c.

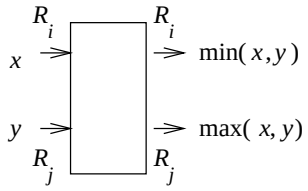


Figure 1: A comparator with the inputs and outputs labeled by R_i and R_j.

We can define a *level* of the comparator c in the network N as follows:

- the level of all comparators with both inputs unconnected is one,
- the level of any comparator with at least one connected input is $l + 1$, where l is the maximal level of the comparators connected to its inputs.

We can divide a computation of a comparator network into *steps*, where during a single step all comparators that have already values on both their inputs compute their outputs. Thus the maximal level is the minimal number of steps needed by the network to compute all its outputs.

For any comparator network N, we can partition its comparators into comparator subsets called *layers*. The sequence of layers $(L_1, \ldots, L_d)$ must satisfy the following conditions:

- If $c \in L_i$ and some input of c is connected to the output of some comparator $c' \in L_k$, then $k < i$.
- For each i, if $c_1 \in L_i$ and $c_2 \in L_i$ and the labels of inputs of c_1 (respectively of c_2) are R_{i_1} and R_{j_1} (respectively R_{i_2} and R_{j_2}) then $\{R_{i_1}, R_{j_1}\} \cap \{R_{i_2}, R_{j_2}\} = \emptyset$.

For a network N with defined sequence of layers $L = (L_1, \ldots, L_d)$ we assume that at step t all comparators from L_t perform its computation. (Even if a comparator of L_t has its inputs already before step t, it waits until step t with its work.) The length of L (i.e. d) is the *depth* of N. Note that the layer L_t can contain only comparators with the level not greater than t.

An equivalent model of the comparator network computation is the following one: The data are stored in the registers labeled by $R_1, \ldots, R_n$, one label per register. During step t, $1 \leq t \leq d$, each comparator c from layer L_t takes the values from the registers R_i and R_j (where R_i and R_j are the labels of the first and the second input of c respectively) and stores the minimum of the two values in the register R_i and the maximum in the register R_j.

Fig. 2 illustrates three styles used for a graphical presentation of a comparator network. The example network contains only four comparators A,B,C and D with the inputs and outputs labeled by the numbers $1, 2, 3, 4$. In the traditional description each comparator is presented as a box with two input lines on its left side and two output lines on its right side. The first input (respectively output) is above the second one. In the "wire-style" each comparator is drawn as

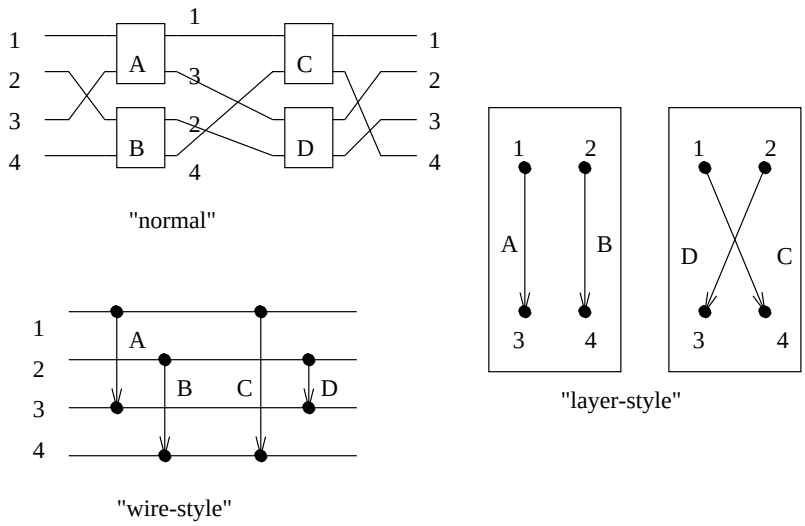


Figure 2: A comparator network presented in different ways

a vertical arrow connecting two wires (each wire corresponds to a single label). The arrow is directed to the wire corresponding to the label of the second (i.e. maximum) output of the comparator. We can draw the vertical lines instead of the arrows if all arrows are directed downwards. In the "layer-style" each layer is drawn as a directed graph of degree 1, where the vertices correspond to the labels and the arcs correspond to the comparators (i.e. if the first and the second output of a comparator c is labeled R_i and R_j respectively, then c is depicted by an arc (R_i, R_j).)

Comparator networks can be easily implemented as specialized hardware devices. This is the main motivation to study these networks. The main parameters of a comparator network are the number of comparators and its depth (the number of layers). The number of comparators corresponds to the amount of hardware needed for the implementation of the network, while its depth corresponds to its computation time. We can also consider other properties of comparator networks such as the complexity of the network architecture and the layout area needed for the implementation of the network in the VLSI technology. The last two issues are extremely important for VLSI design and suitability for practical applications. One nontrivial issue is to judge what "simple architecture" is. There is no easy way to express it in a mathematical model so that all technological limitations are well modeled. For this reason we mainly discuss such parameters as depth and size of comparator networks, while quality of architecture is often expressed in an intuitive way.

Comparator networks has been the subject of intensive investigations in computer science. Their main application is sorting input sequences. But there are also many other tasks that can be performed by the comparator networks. For example we can use them for

- merging sorted subsequences into a single sorted sequence,
- sorting sequences that differ from a sorted sequence only on a limited number of positions,
- inserting a value into a sorted sequence, so that the output is sorted,
- selecting the minimal or the maximal value (or the t smallest or the t greatest values) of the input.

The first three applications can be considered as sorting of the constrained input sequences, since the output must be sorted. All these applications gain a growing interest due to the needs in telecommunication technology. Efficient methods of packet reordering may provide new designs of intelligent routers and similar devices. Since communication bottleneck is one of most severe problems in computer technology and practice today, these methods deserve a lot of attention.

There is a lower bound of $\Omega(\log n)$ on the depth of the comparator networks for all of the above listed problems, where n is the size of the input. On the other hand, there is a sorting network of depth $O(\log n)$, known as the AKS network [1]. Since these problems are less general than the problem of sorting, the upper bound on the depth is also $O(\log n)$. The best currently known sorting networks of depth $O(\log n)$ (which are variants of the AKS network) have the depth not less than $c \log n$, where c is a constant not less than 1000. Additionally, their architecture is very complex and is based on the structure of (usually random) expander graphs or other random structures. Thus, for practical applications we have to find other networks. The most elegant and the most efficient practical designs are the two Batcher networks [3]. They have the depth very close to $\frac{1}{2}\log^2 n$ and are based respectively on the odd-even and bitonic merging networks. Note that the Batcher networks beat the AKS network for the inputs of size $n \leq 2^{1000}$. Since 2^{1000} is much bigger than the estimated number of the particles in the universe, for any potential application the AKS network is inferior to the Batcher networks.

1.2 Periodic networks

We can also consider so called *periodic* networks. Periodic networks perform their computation in many iterations. During each iteration a sequence stored in the registers is taken as an input configuration of the network and is replaced by the output configuration computed by the network for this input. Thus although the computation time is $t = dk$, where k is the number of iterations and d is the depth of the network, the number of comparators is at most $dn/2$, where n is the input size (since each layer can contain at most $\lfloor n/2 \rfloor$ comparators). The examples of the periodic networks are

- the DPSR network by M. Dowd, Y. Perl, M. Saks, and L. Rudolph [5], of depth $\log n$ that sorts in $\log n$ iterations,

- the network by M. Kutyłowski, K. Loryś, B. Oesterdiekhoff, and R. Wanka [10] of a constant depth that sorts in $O(\log^2 n)$ iterations (obtained by so called periodification of the AKS network),
- the odd-even transposition network of depth 2 that sorts in $n/2$ iterations,
- the network by I. D. Scherson, S. Sen, and A. Shamir [14] of depth $2\sqrt{n}$ that sorts in $\log n$ iterations,
- the Schwiegelshohn network [16] of depth 8 that sorts in $O(\sqrt{n}\log n)$ iterations.

The last three networks are very suitable for the VLSI technology, since the area of the layout of their underlying architecture is proportional to the size of the input. There are also known periodic merging networks of a constant depth by M. Kutyłowski, K. Loryś, B. Oesterdiekhoff [9] that merge two sequences in $O(\log n)$ iterations.

1.3 Outline of the thesis

In this thesis the following results are presented:

- In Section 3, for an arbitrary constant k we present a periodic network of a constant depth that sorts in $O(n^{1/k})$ iterations. The construction of this network is based on the (ε, m)-*blocks* which are a generalization of the odd-even transposition network, where single registers are replaced by the groups of m registers, and comparators are replaced by the so called ε-halvers (used originally in the construction of the AKS network). This network (presented in [8]) was asymptotically the best constant depth network until the invention of the networks from [10].
- In Section 4, we present a comparator network that sorts any sequence that differs from some sorted sequence at at most k positions (a so called k-*disturbed* sequence). The depth of the network is $4\log n + O(\log^2 k \log\log n)$, and hence for $k = o\left(2^{\sqrt{\log n/\log\log n}}\right)$, the depth of the network is $4\log n + o(\log n)$. Thus the constant in front of $\log n$ is much smaller than the constant in the asymptotically optimal AKS sorting network.
- Section 5 presents a periodic correction network of a constant depth that sorts any k-disturbed sequence of length n in $O(\log n + k)$ iterations.

Results of Section 3 are due to M. Kutyłowski, G. Stachowiak and myself. These results have been published in [8].

The result of Section 4 is a refinement of the construction from [6] and has been been published as a joint work by M. Kutyłowski, M. Piotrów and myself in [7].

The results of the Section 5 are unpublished yet. They are inspired by the idea presented by Grzegorz Stachowiak of adding the back-jump comparators to the network H_l of the Subsection 5.1.

2 Preliminaries

In this section we introduce formal definitions of the basic concepts used in the remaining part of the thesis. We also present here some simple but useful lemmas that simplify the analysis of the comparator network computations.

Many constructions, definitions and proofs in this thesis might be regarded at first by the reader as too formal. However, comparator networks require very strict and precise definitions, since in many cases even small changes in their constructions may cause serious deterioration of their performance.

We assume that all logarithms (unless stated otherwise) are to the base of two.

Definition 2.1 *Let $P = (P_1, \ldots, P_k)$ and $Q = (Q_1, \ldots, Q_l)$ be two arbitrary sequences. By PQ we denote the sequence $(P_1, \ldots, P_k, Q_1, \ldots, Q_l)$ (*concatenation *of P and Q). By P^0 we denote an empty sequence, and for $i > 0$, P^i denotes $P^{i-1}P$.*

Definition 2.2 *Let X be a finite ordered set. Let $x \in X$. Then the* rank *of x in X is a positive integer r such that $r = |\{y \in X \mid y \leq x\}|$.*

In the following definitions we formalize the notion of comparator network introduced in Section 1. We will identify the registers by their labels that are integer numbers. The comparator is identified by the pair of registers that it compares and the layer is a subset of comparators.

Definition 2.3 *Let R be any finite subset of positive integers. We call a subset L of $R \times R$ a* layer over R *if and only if:*

- *for each $(i, j) \in L$, $i \neq j$, and*
- *each element of R is contained by at most one ordered pair in L.* [1]

The elements of the layers are called comparators.

Definition 2.4 *Let n and d be any positive integers. Let R be a set of n integers (we call them* registers*). Let $L = (L_1, L_2, \ldots, L_d)$ be a sequence of layers over S. Then by $CN(n, d, R, L)$ we denote the* comparator network *of* input size n, depth d *on the* set of registers R *with the* sequence of layers L. *For $1 \leq i \leq n$, by R_i we denote the element of R with a rank i (i.e. the ith register of R).*

Definition 2.5 *Let X be any ordered set. Let R be a set of n registers. Then any function $c : R \to X$ is called a* configuration of R over X. *The sequence $(c(R_1), \ldots, c(R_n))$ is called a* configuration sequence of R. *We say that the register R_i* contains *the value $c(R_i)$. For any subset of registers $S' \subseteq \{R_1, \ldots, R_n\}$ for any $x \in X$, we say that c* has k values x in S' *if and only if $|\{R_i \in S' \mid c(R_i) = x\}| = k$.*

Let $S \subseteq \{R_1, \ldots, R_n\}$. We call a configuration $c' : S \to X$ a S-restriction *of c if and only if $c'(R_i) = c(R_i)$ for each $R_i \in S$.*

[1] In another context we call such sets *matchings*.

Definition 2.6 *Let X be any ordered set. Let R be a set of n registers. Let c be any configuration of R over X. Let L be a layer over $\{R_1, \ldots, R_n\}$. Then by $L(c)$ we denote the configuration c' of R over X obtained after executing comparators from L:*

- *for each $(i,j) \in L$, $c'(i) = \min\{c(i), c(j)\}$ and $c'(j) = \max\{c(i), c(j)\}$ (we say that comparator (i,j)* compares *the registers i and j in the layer L),*
- *for each $r \in R$ such that there is no pair containing r in L, $c'(r) = c(r)$.*

We call $L(c)$ a result of application of L on c.

Definition 2.7 *Let X be an ordered set. Let R be a set n of registers. Let $L = (L_1, \ldots, L_d)$ be a sequence of layers over R. Let c be any configuration of R over X. For $0 \leq i \leq d$ we define a sequence of configurations $L(i,c)$ as follows:*

- *$L(0,c) = c$, and*
- *for $1 \leq i \leq d$, $L(i,c) = L_i(L(i-1,c))$.*

We call the sequence $(L(0,c), \ldots, L(d,c))$ a computation trace of L on c. *We also use $L(c)$ to denote $L(d,c)$.*

The following definitions introduce notations used for constructing new networks from already defined layers.

Definition 2.8 *Let S and S' be two finite subsets of positive integers such that $|S| = |S'|$. Let f be any bijection between S and S'. Let L be a layer over S. Then the f-*mapping of L *is the layer L' over S' defined as follows:*

$$L' = \{(f(i), f(j)) \mid (i,j) \in L\}.$$

If L is a sequence of layers $(L_1, \ldots, L_d)$ over S, then the f-mapping of L is a sequence $L' = (L'_1, \ldots, L'_d)$, where L'_i is an f-mapping of L_i.

Definition 2.9 *Let S and S' be any finite subsets of positive integers. Let L be a layer over S. Then the S'-*restriction of L *is the layer L' over S' defined as follows:*

$$L' = \{(i,j) \in L \mid i,j \in S'\}.$$

If L is a sequence of layers $(L_1, \ldots, L_d)$ over S, then the S'-restriction of L is a sequence $L' = (L'_1, \ldots, L'_d)$, where L'_i is an S'-restriction of L_i.

Definition 2.10 *Let S and S' be two subsets of registers. Let $d \geq 1$. Let $L = (L_1, \ldots, L_d)$ and $L' = (L'_1, \ldots, L'_d)$ be the sequences of layers over S and S' respectively, such that each $L_i \cup L'_i$ is a layer over $S \cup S'$. Then the union of L and L' (denoted by $L \cup L'$) is the sequence of layers $(L_1 \cup L'_1, \ldots, L_d \cup L'_d)$.*

Definition 2.11 *A comparator (r_1, r_2) is called a* standard *comparator if and only if $r_1 < r_2$. A layer L is called a* standard *layer if and only if it contains only standard comparators. A network $CN(n, d, R, L)$ is a* standard *network if and only if L is the sequence of standard layers.*

All the comparator networks considered in the following sections are standard networks.

We will frequently use the following trivial but useful observation.

Lemma 2.12

- *If L is a standard layer, then any S-restriction of L is also a standard layer.*
- *If the union of standard layers is a layer, then it is a standard layer.*
- *If L is a standard layer over S and $f : S \to S'$ is an increasing one to one function, then the f-mapping of L is a standard layer over S'.*

The following simple lemma and corollary state that we can clip a standard network to an arbitrary size preserving many of its properties.

Lemma 2.13 *Let R be a set of n registers. Let $m \leq n$ and $R' = \{R_1, \ldots, R_m\}$. Let L be a standard layer over R and let L' be the R'-restriction of L. Let c be a configuration of R such that $c(R_i) = c_{\max}$, for each $i > m$, where $c_{\max} = \max\{c(r) | r \in R\}$. Let c' be the R'-restriction of c. Then*

1. *$L'(c')$ is an R'-restriction of $L(c)$, and*
2. *$L(c)(R_i) = c_{\max}$ for each $i > m$.*

Proof. Let $r \in R'$. If there is no comparator in L containing r, then $L'(c')(r) = c'(r) = c(r) = L(c)(r)$. If there is $r' \in R'$ such that

$$(\min\{r, r'\}, \max\{r, r'\}) \in L,$$

then

$$(\min\{r, r'\}, \max\{r, r'\}) \in L'$$

and hence $L'(c')(r) = L(c)(r)$. If there is $r' \in R \setminus R'$ such that $(r, r') \in L$, then $c(r') = c_{\max}$ and hence $L'(c')(r) = c(r) = L(c)(r)$.

The second property follows immediately from the fact that for $i > m$, $c(R_i) = c_{\max}$ and that any register connected by a comparator to R_i must either be the first register of the comparator or contain also the value $c_{\max}$ in configuration c. □

An immediate consequence of Lemma 2.13 is the following useful corollary.

Corollary 2.14 *Let R, R', c, c' and $c_{\max}$ be defined as in Lemma 2.13. Let L be a sequence of standard layers over S, let L' be an S'-restriction of L. Then*

1. *$L'(c')$ is an R' restriction of $L(c)$, and*

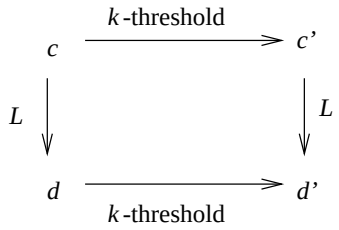


Figure 3: Lemma 2.18.

2. $L(c)(R_i) = c_{\max}$ *for each* $i > m$.

Definition 2.15 *Let X be an ordered set and let R be a set of n registers. A network $CN(n, d, R, L)$* sorts *a configuration c of R (and the sequence $(c(R_1), \ldots, c(R_n))$) if and only if the sequence $(L(c)(R_1), \ldots, L(c)(R_n))$ is a nondecreasing sequence.*

A network $N = CN(n, d, R, L)$ is called a sorting network over X *if and only if for all configurations $c : R \to X$, network N sorts c.*

Any sorting network can be transformed into a standard sorting network of the same depth and with the same number of comparators in each layer (see exercise 16 on page 239 in [11]).

The following lemma from [11], called Zero-One Principle and is a fundamental tool for analyzing comparator networks.

Lemma 2.16 *A comparator network N is a sorting network over any ordered set X if and only if N is a sorting network over $\{0, 1\}$.*

We call the configurations over $\{0, 1\}$ *zero-one configurations.*

Zero-One Principle is a consequence of a slightly more general fact, which we present below.

Definition 2.17 *Let X be an ordered set. Let R be a set of n registers. Let c be a configuration of R over X and let Y be a set of values of c. For integer k a* k-threshold *of c is a configuration c' of R over $\{0, 1\}$ such that:*

$$c'(R_i) = \begin{cases} 0 & \text{if rank of } c(R_i) \text{ in } Y \text{ is less than } k, \\ 1 & \text{otherwise.} \end{cases}$$

Lemma 2.18 *Let X be an ordered set. Let R be a set of n registers. Let c be a configuration of R over X. Let k be any integer and let c' be a k-threshold of c. Let L be a layer over R. Let $d = L(c)$ and let d' be a k-threshold of d. Then $d' = L(c')$. (See Fig. 3.)*

Proof. Let Y be a set of values of c. (Y is also a set of values of d.) Suppose that $d' \neq L(c')$. Then there is an index i such that $d'(R_i) \neq L(c')(R_i)$, that is, either $d'(R_i) = 0$ and $L(c')(R_i) = 1$ or $d'(R_i) = 0$ and $L(c')(R_i) = 1$. In the first case the rank of $d(R_i) = L(c)(R_i)$ in Y is less than k. But $L(c')(R_i) = 1$ implies

that $c'(R_i) = 1$ or there is a comparator (R_j, R_i) in L where $c'(R_j) = 1$. Thus the rank of $c(R_i)$ in Y is at least k or $L(c)(R_i)$ is maximum of the two values, with at least one of them having the rank greater or equal k. Hence the rank of $L(c)(R_i)$ must be greater or equal k. Contradiction. The case $d'(R_i) = 0$ and $L(c')(R_i) = 1$ is analogous. □

Lemma 2.19 *Let X be an ordered set. Let R be a set of n registers. Let c be a configuration of R over X and Y be the set of values of c. For $1 \leq k \leq n$, let c_k be k-threshold of c. Then there is no other configuration c' with the set of values Y such that for each k, c_k is a k-threshold of c'.*

Proof. Suppose that there is such a configuration c', $c' \neq c$. Then $c(R_i) \neq c'(R_i)$, for some i. The rank r of $c(R_i)$ is different from the rank r' of $c'(R_i)$ in Y, since Y is an ordered set. Consider the case $r < r'$ (equivalent to $r+1 \leq r'$). Then $c_{r+1}(R_i) = 0$ which is a contradiction to c_{r+1} being a $(r+1)$-threshold of c'. The case $r' < r$ is analogous. □

Corollary 2.20 *Let X, R, c, Y and c_k be defined as in Lemma 2.19. Then there is no other configuration c' with the set of values Y such that for each $2 \leq k \leq |Y|$, c_k is a k-threshold of c'.*

Proof. It follows from Lemma 2.19 and from the fact that c_1 is a constant function equal to 1 and, for $k > |Y|$, configurations c_k are constant functions equal to 0, and the constant k-thresholds do not impose any restrictions on the configuration. □

Lemma 2.21 *Let X be an ordered set. Let R be a set of n registers. Let c and c' be configurations of R over X with the same set of values and with each value occuring as many times in c as in c'. For $1 \leq k \leq n$, let c_k (respectively c'_k) be a k-threshold of c (respectively c'). Let L be any layer over R. Then $c' = L(c)$ if and only if for each k, $1 \leq k \leq n$, $c'_k = L(c_k)$.*

Proof. If $c' = L(c)$, then by Lemma 2.18, for each k, $c'_k = L(c_k)$. If for each k, $1 \leq k \leq n$, $c'_k = L(c_k)$, then by the fact that each $L(c_k)$ is a k-threshold of $L(c)$ and by Lemma 2.19 we have $c' = L(c)$. □

The following lemma is a simple but useful modification of the zero-one principle.

Lemma 2.22 *Let X be an ordered set. Let R be a set of n registers. Let c and c' be configurations of R over X. For $1 \leq k \leq n$, let c_k (respectively c'_k) be a k-threshold of c (respectively c'). Let $L = (L_1, \ldots, L_d)$ be a sequence of layers over R. Then $c' = L(d, c)$ if and only if for each k, $1 \leq k \leq n$, $c'_k = L(d, c_k)$.*

Proof. Immediate consequence of Lemma 2.21 □

The following definition is specific for the zero-one configurations.

Definition 2.23 *Let R be a set of n registers. Let c be a configuration of R over $\{0,1\}$. Let $p > 0$. We say that c is* p-dirty *if and only if there is an index i such that for all $1 \leq j \leq i-1$, $c(R_j) = 0$ and for all $i+p \leq j \leq n$, $c(R_j) = 1$. The subset of registers that are between the first register containing one and the last register containing zero is called* dirty region *of c.*

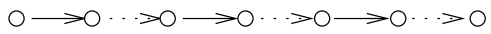


Figure 4: The n-odd-even transposition network for $n = 7$. The first layer is drawn with solid lines and the second layer is drawn with dashed lines.

3 Periodic Sorting Networks

In this section for arbitrary constant k, we present a periodic network of a constant depth that sorts a sequence of n elements in $O(n^{1/k})$ iterations. The construction of the network is based on the so called (ε, m)-blocks. An (ε, m)-block is a generalization of the well known odd-even transposition sorting network where the registers are replaced by the groups of m registers and the comparators are substituted by the so called ε-halvers.

3.1 Periodic Networks. Preliminaries

Recall that a periodic network (Section 1.2) processes the data stored in registers in many iterations. Here we specify more formally the notion of a periodic sorting network.

Definition 3.1 *Let N be a comparator network $CN(n, d, R, L)$. N is a* periodic sorting network *if and only if for some positive integer t the network $CN(n, td, R, L^t)$ is a sorting network. We say that N sorts in t iterations.*

3.1.1 Odd-even transposition network

The simplest periodic sorting network is the following one:

Definition 3.2 *Let $R = \{1, 2, \ldots, n\}$. An* n-odd-even transposition *network (see Fig. 4) is a network $CN(n, 2, R, (L_1, L_2))$, where*

$$L_1 = \{(i, i+1) \mid \ 1 \leq i < n \text{ and } i \text{ is odd}\}$$

and

$$L_2 = \{(i, i+1) \mid \ 2 \leq i < n \text{ and } i \text{ is even}\}.$$

Let us recall the following well known facts (see [11] and [4]):

Lemma 3.3 *[11] The n-odd-even transposition network sorts in $\lceil n/2 \rceil$ iterations.*

Lemma 3.4 *[4] If a periodic standard network N on registers $\{1, \ldots, n\}$ contains all comparators of the n-odd-even transposition network, then for each $k \leq n$, N sorts each k-dirty configuration in at most k iterations.*

3.1.2 ε-halvers

In the construction of our periodic networks we will use comparator networks called ε-halvers. The notion of ε-halver was introduced in [1]. Informally, halving is the task of moving the greater values to the second half of the registers and the smaller values to the first half of registers, so that no value in the first half is bigger than any value in the second half. The ordering of the values inside a half does not matter. If we consider only zero-one configurations of the set of registers R of size $2m$ with x_0 zeroes and x_1 ones, then the halver either moves all the ones to $R_{m+1}, \ldots, R_{2m}$ or all the zeroes to $R_1, \ldots, R_m$. The exact halver must have a depth $\Omega(\log m)$. This follows from the Alekseyev's lower bound $(n-t)\lceil \log(t+1) \rceil$ on the number of comparators for selecting $t = m$ smallest elements in the sequence of length $n = 2m$. (See [11], page 234.)

Ajtai, Komolos and Szemeredi [1] introduced so called ε-halvers. The difference between halvers and ε-halvers is that in the later case we demand that ε-halver leaves either at most εx_0 zeroes in $R_{m+1}, \ldots, R_{2m}$ (if $x_0 \leq x_1$), or at most εx_1 ones in $R_{m+1}, \ldots, R_{2m}$ (if $x_1 \leq x_0$) instead of moving all zeroes or ones to the proper half. It is surprising that, for $\varepsilon > 0$, there exist ε-halvers of the depth independent on the number of their registers. This led to construction of the famous AKS network. The construction of ε-halvers is based on the random bipartite graphs called *expanders*. The tradeoff between the depth of ε-halver and the value ε, and their random structure, make the networks based on ε-halvers rather impractical. However we use them in our construction to obtain good asymptotical estimation of the runtime.

Below we introduce a more formal definition of an ε-halver. First we define auxiliary functions:

Definition 3.5 *Let $\varepsilon \in [0, \frac{1}{2})$ and let $m > 0$. We define two functions over $[0, 2m]$ (see Fig. 5):*

$$f_{\varepsilon,m}(x) = \begin{cases} \varepsilon x & \text{for } x \leq m, \\ m - (1-\varepsilon)(2m - x) & \text{for } x > m, \end{cases}$$

$$g_{\varepsilon,m}(x) = \begin{cases} (1-\varepsilon)x & \text{for } x \leq m, \\ m - \varepsilon(2m - x) & \text{for } x > m. \end{cases}$$

Let us state the following obvious properties:

Lemma 3.6 *$f_{\varepsilon,m}$ is a convex function. The functions $f_{\varepsilon,m}$ and $g_{\varepsilon,m}$ are non-decreasing, continuous, and for $0 \leq x \leq 2m$ the following holds:*

- $0 \leq f_{\varepsilon,m}(x) \leq m$, $0 \leq g_{\varepsilon,m}(x) \leq m$,
- $f_{\varepsilon,m}(x) \leq x$, $g_{\varepsilon,m}(x) \leq x$,
- $f_{\varepsilon,m}(x) + g_{\varepsilon,m}(x) = x$,
- $f_{\varepsilon,m}(x) = m - g_{\varepsilon,m}(2m - x)$,
- $g_{\varepsilon,m}(x) = m - f_{\varepsilon,m}(2m - x)$.

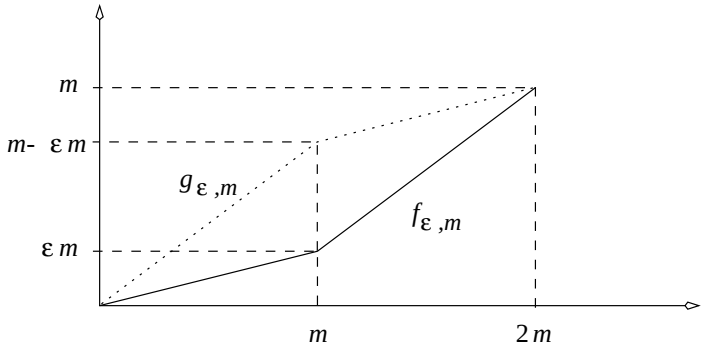


Figure 5: The functions $f_{\varepsilon,m}$ and $g_{\varepsilon,m}$

Definition 3.7 *Let $\varepsilon \geq 0$. Let n be an even positive integer. Let $R = \{1, \ldots, n\}$ be a set of registers. A comparator network $N = CN(n, d, R, L)$ is an* ε-halver *on R if the following holds. For each configuration c of R over $\{0, 1\}$ such that $|\{r \in R \mid c(r) = 1\}| = x$ the configuration $c' = L(d, c)$ has the following properties:*

- $|\{r \in R \mid r \leq n/2,\ c'(r) = 1\}| \leq f_{\varepsilon,n/2}(x)$, *and*
- $|\{r \in R \mid r > n/2,\ c'(r) = 1\}| \geq g_{\varepsilon,n/2}(x)$.

Note that by the last equality stated in Lemma 3.6 ε-halver is symmetrical in the following sense:

Lemma 3.8 *Let N be a ε-halver on $R = \{1, \ldots, n\}$ registers for some even $n > 0$. Let c' be an output configuration of N for some input zero-one configuration c such that $|\{r \in R \mid c(r) = 0\}| = x$. Then:*

- $|\{r \in R \mid r > n/2,\ c'(r) = 0\}| \leq f_{\varepsilon,n/2}(x)$, *and*
- $|\{r \in R \mid r \leq n/2,\ c'(r) = 0\}| \geq g_{\varepsilon,n/2}(x)$.

The following lemma is due to Ajtai, Komolos and Szemeredi and states the key property of ε-halvers:

Lemma 3.9 *[1] For each $\varepsilon > 0$ there exist a constant positive integer d_ε, such that for each even positive integer n, there is an ε-halver on $\{1, \ldots, n\}$ of depth d_ε.*

3.1.3 (ε, m)-blocks

Below we use ε-halvers to define (ε, m)-blocks that are the basic elements used in the construction of our network. An (ε, m)-block can be presented as the odd-even transposition network, where each register is replaced by a group of m registers, and each comparator is replaced by an ε-halver on the corresponding pairs of groups.

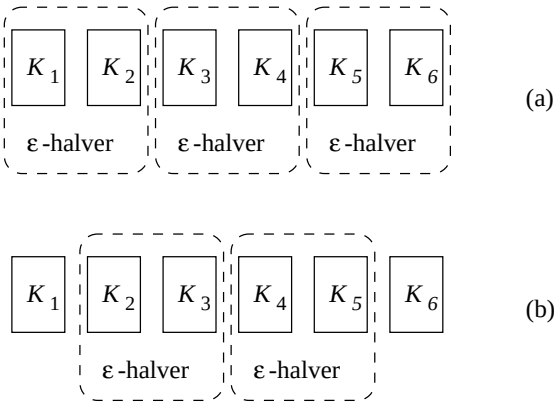


Figure 6: The upper (a) and the lower (b) part of an (ε, m, n)-block, for $n = 6m$.

Definition 3.10 *Let $\varepsilon > 0$. Let m and n be positive integers such that $n = km$ for some integer $k \geq 2$. Let $R = \{1, \ldots, n\}$. Let $N = CN(2m, d_\varepsilon, \{1, \ldots, 2m\}, L)$ be an ε-halver. For $1 \leq j \leq k$ let $K_j = \{(j-1)m + 1, \ldots, jm\}$ and let $f_j : \{1, \ldots, 2m\} \to K_j \cup K_{j+1}$ be a function such that $f_j(i) = (j-1)m + i$. We call a network $M = CN(n, 2d_\varepsilon, R, L_1 L_2)$ an (ε, m, n)-*block *(or shortly (ε, m)-block on R) if and only if for each t, $1 \leq t \leq d_\varepsilon$ (see Fig. 6):*

- *L_1 is the union of the f_j-mappings of L for all odd j, $1 \leq j < k$,*
- *L_2 is the union of the f_j-mappings of L for all even j, $2 \leq j < k$.*

We call the subsequence of layers L_1 an upper part *of the (ε, m, n)-block and the subsequence of layers L_2 a* lower part *of the (ε, m, n)-block. We call the subset K_j the* jth m-bucket *of R.*

If n is not a multiple of m, then by (ε, m, n)-block we mean a network $CN(n, 2d_\varepsilon, (1, \ldots, n), L')$, where L' is the $\{1, \ldots, n\}$-restriction of the sequence of layers of the $(\varepsilon, m, \lceil n/m \rceil m)$-block.

Note that according to Definition 3.10 the n-odd-even transposition network is an $(0, 1, n)$-block but its depth is 2 instead of d_ε.

3.2 Properties of the (ε, m)-blocks

In this section we prove Lemma 3.23, key property of (ε, m)-blocks. It states that an (ε, m, n)-block shrinks the dirty region to $O(m \log n)$ registers in $O(n/m)$ iterations. We commence with some auxiliary lemmas and definitions.

Definition 3.11 *Let $a = (a_1, \ldots, a_l)$ be a vector of real numbers. For $1 \leq i \leq l$, $\mathrm{hd}_i(a)$ (a* head *of a) denotes the prefix sum $\sum_{j=1}^{i} a_j$. (We assume that $\mathrm{hd}_i(a) = 0$ for $i < 1$ and $\mathrm{hd}_i(a) = \mathrm{hd}_l(a)$ for $i > l$). For $1 \leq i \leq l$, $\mathrm{tl}_{m,i}(a)$ (a* tail *of a) denotes the sum $\sum_{j=i}^{l}(m - a_j)$. (We assume that $\mathrm{tl}_{m,i}(a) = 0$ for $i > l$, and $\mathrm{tl}_{m,i}(a) = \mathrm{tl}_{m,1}(a)$ for $i < 1$).*

Definition 3.12 *Let $a = (a_1, \ldots, a_l)$, $b = (b_1, \ldots, b_l)$, where $a_i, b_i \in [0, m]$ for $1 \leq i \leq l$. We say that b* dominates *a (denoted by $a \preceq b$) if and only if $\mathrm{hd}_l(a) = \mathrm{hd}_l(b)$ and $\mathrm{hd}_k(b) \leq \mathrm{hd}_k(a)$, for every k, $1 \leq k \leq l$.*

Note that $a \preceq b$ if and only if $\mathrm{tl}_{m,1}(a) = \mathrm{tl}_{m,1}(b)$ and $\mathrm{tl}_{m,k}(b) \leq \mathrm{tl}_{m,k}(a)$ for every $1 \leq k \leq l$.

The following properties follow directly from the definition.

Lemma 3.13 *The relation $\preceq$ is a partial order on $[0, m]^l$.*

Lemma 3.14 *Let L be any sequence of standard layers on $R = \{1, \ldots, lm\}$. Let c be a configuration of R over $\{0, 1\}$. Let $c' = L(c)$. Let K_i denote the ith m-bucket of R. Let $x = (x_1, \ldots, x_l)$ and $y = (y_1, \ldots, y_l)$ be two vectors such that for each i, $1 \leq i \leq l$,*

- *$x_i = |\{j \mid j \in K_i,\ c(j) = 1\}|$ and*
- *$y_i = |\{j \mid j \in K_i,\ c'(j) = 1\}|$.*

Then $x \preceq y$.

Definition 3.15 *Let $\varepsilon \geq 0$. Let $m > 0$. Let $a = (a_1, \ldots, a_l)$ be a vector of real numbers such that $a_i \in [0, m]$ for all i. By $\mathcal{N}_{\varepsilon,m}(a)$ we denote a vector $x = (x_1, \ldots, x_l)$ such that for each odd j, $1 \leq j < l$:*

- *$x_j = f_{\varepsilon,m}(a_j + a_{j+1})$, and*
- *$x_{j+1} = g_{\varepsilon,m}(a_j + a_{j+1})$ and*
- *if l is odd, then $x_l = a_l$.*

By $\mathcal{P}_{\varepsilon,m}(a)$ we denote a vector $x = (x_1, \ldots, x_l)$ such that for each even j, $2 \leq j < l$:

- *$x_1 = a_1$, and*
- *$x_j = f_{\varepsilon,m}(a_j + a_{j+1})$, and*
- *$x_{j+1} = g_{\varepsilon,m}(a_j + a_{j+1})$ and*
- *if l is even, then $x_l = a_l$.*

Let us comment the above definition. Let $a = (a_1, \ldots, a_l)$ be a sequence such that a_i is the number of ones in K_i. Assume that an (ε, m)-block executes its upper part. Consider the number of ones that remain in K_i for i odd. By the definition of an ε-halver, it is upper bounded by $f_{\varepsilon,m}(a_i + a_{i+1})$. The number of ones in K_{i+1} is at that moment at least $g_{\varepsilon,m}(a_i + a_{i+1})$. So we may regard $\mathcal{N}_{\varepsilon,m}(a)$ as a pessimistic estimate on the placement of ones.

Lemma 3.16 *Let x_1 and x_2 be two vectors from $[0, m]^l$ such that, for $1 \leq i \leq l$, $\mathrm{hd}_i(x_1) \leq \mathrm{hd}_i(x_2)$. Then, for $1 \leq i \leq l$, $\mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x_1)) \leq \mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x_2))$ (respectively $\mathrm{hd}_i(\mathcal{P}_{\varepsilon,m}(x_1)) \leq \mathrm{hd}_i(\mathcal{P}_{\varepsilon,m}(x_2))$).*

Proof. If i is even, then $\mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x_1)) = \mathrm{hd}_i(x_1)$ and $\mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x_2)) = \mathrm{hd}_i(x_2)$. If i is odd, then

$$\begin{aligned}\mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x_1)) &= \mathrm{hd}_{i-1}(x_1) + f_{\varepsilon,m}(x_{1,i} + x_{1,i+1}) \\ &= \mathrm{hd}_{i+1}(x_1) - g_{\varepsilon,m}(x_{1,i} + x_{1,i+1})\end{aligned}$$

and

$$\begin{aligned}\mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x_2)) &= \mathrm{hd}_{i-1}(x_2) + f_{\varepsilon,m}(x_{2,i} + x_{2,i+1}) \\ &= \mathrm{hd}_{i+1}(x_2) - g_{\varepsilon,m}(x_{2,i} + x_{2,i+1}).\end{aligned}$$

If $x_{1,i} + x_{1,i+1} \leq x_{2,i} + x_{2,i+1}$, then by the fact that

$$\mathrm{hd}_{i-1}(\mathcal{N}_{\varepsilon,m}(x_1)) \leq \mathrm{hd}_{i-1}(\mathcal{N}_{\varepsilon,m}(x_2))$$

and that $f_{\varepsilon,m}$ is nondecreasing function we have

$$\mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x_1)) \leq \mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x_2)).$$

If $x_{1,i} + x_{1,i+1} > x_{2,i} + x_{2,i+1}$ then by the fact that

$$\mathrm{hd}_{i+1}(\mathcal{N}_{\varepsilon,m}(x_1)) \leq \mathrm{hd}_{i+1}(\mathcal{N}_{\varepsilon,m}(x_2))$$

and that $g_{\varepsilon,m}$ is nondecreasing function we have

$$\mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x_1)) \leq \mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x_2)).$$

The proof of the claim for $\mathcal{P}_{\varepsilon,m}$ is analogous.

Lemma 3.17 *Let x and y be two vectors from $[0,m]^l$ such that $x \preceq y$. Then $\mathcal{N}_{\varepsilon,m}(x) \preceq \mathcal{N}_{\varepsilon,m}(y)$ and $\mathcal{P}_{\varepsilon,m}(x) \preceq \mathcal{P}_{\varepsilon,m}(y)$.*

Proof. The lemma follows directly from Claim 3.16. □

Lemma 3.18 *Let $R = \{1, \ldots, lm\}$ be a sequence of registers. Let L be an upper (respectively a lower) part of the (ε, m)-block on R. Let c be any configuration of R over $\{0,1\}$. Let $c' = L(c)$. Let $x = (x_1, \ldots, x_l)$ and $y = (y_1, \ldots, y_l)$ be vectors such that, for each i, $1 \leq i \leq l$,*

$$x_i = |\{j \mid j \in K_i,\ c(j) = 1\}|$$

and

$$y_i = |\{j \mid j \in K_i,\ c'(j) = 1\}|,$$

where K_i is ith m-bucket of R. Let $x' \in [0,m]^l$ be any vector such that $x' \preceq x$. Then $\mathcal{N}_{\varepsilon,m}(x') \preceq y$ (respectively $\mathcal{P}_{\varepsilon,m}(x') \preceq y$).

Proof. We prove the lemma only for the upper part of the (ε, m)-block and $\mathcal{N}_{\varepsilon,m}$. The lemma for the lower part and $\mathcal{P}_{\varepsilon,m}$ is analogous. By Lemma 3.17, $\mathcal{N}_{\varepsilon,m}(x') \preceq \mathcal{N}_{\varepsilon,m}(x)$. Thus it is sufficient to show that $\mathcal{N}_{\varepsilon,m}(x) \preceq y$. If i is even or $i = l$, then $\mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x)) = \mathrm{hd}_i(x) = \mathrm{hd}_i(y)$. If i is odd and $i < l$, then $\mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x)) = \mathrm{hd}_{i-1}(x) + f_{\varepsilon,m}(x_i + x_{i+1}) = \mathrm{hd}_{i+1}(x) - g_{\varepsilon,m}(x_i + x_{i+1})$. On the other hand, by the fact that there is ε-halver on the $K_i \cup K_{i+1}$ in the upper part of (ε, m)-block,

$$\mathrm{hd}_i(y) \leq \mathrm{hd}_{i-1}(y) + f_{\varepsilon,m}(y_i + y_{i+1}) = \mathrm{hd}_{i+1}(y) - g_{\varepsilon,m}(y_i + y_{i+1}).$$

Consider the cases $x_i + x_{i+1} \geq y_i + y_{i+1}$ and $x_i + x_{i+1} < y_i + y_{i+1}$, in a similar way as in the proof of Lemma 3.16:

- If $x_i + x_{i+1} \geq y_i + y_{i+1}$, then $\mathrm{hd}_i(y) \leq \mathrm{hd}_{i-1}(y) + f_{\varepsilon,m}(y_i + y_{i+1}) \leq \mathrm{hd}_{i-1}(x) + f_{\varepsilon,m}(x_i + x_{i+1}) = \mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x))$.
- If $x_i + x_{i+1} < y_i + y_{i+1}$, then $\mathrm{hd}_i(y) \leq \mathrm{hd}_{i+1}(y) - g_{\varepsilon,m}(y_i + y_{i+1}) \leq \mathrm{hd}_{i+1}(x) - g_{\varepsilon,m}(x_i + x_{i+1}) = \mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x))$.

Thus $\mathrm{hd}_i(y) \leq \mathrm{hd}_i(\mathcal{N}_{\varepsilon,m}(x))$. □

Definition 3.19 *Let $\varepsilon \geq 0$. Let $m > 0$. Let $x \in [0, m]^l$. For integers $t \geq 0$ we define a sequence $\mathcal{V}^t_{\varepsilon,m}(x)$ as follows:*

- $\mathcal{V}^0_{\varepsilon,m}(x) = x$,
- $\mathcal{V}^t_{\varepsilon,m}(x) = \mathcal{N}_{\varepsilon,m}(\mathcal{V}^{t-1}_{\varepsilon,m}(x))$ *for odd* $t \geq 1$,
- $\mathcal{V}^t_{\varepsilon,m}(x) = \mathcal{P}_{\varepsilon,m}(\mathcal{V}^{t-1}_{\varepsilon,m}(x))$ *for even* $t \geq 2$.

(If t is not integer, then by $\mathcal{V}^t_{\varepsilon,m}(x)$ we denote $\mathcal{V}^{\lceil t \rceil}_{\varepsilon,m}(x)$.)

Let x be the minimal vector in $[0, m]^l$ with the sum of coordinates equal to the number of ones in some initial zero-one configuration. We use the values $\mathcal{V}^t_{\varepsilon,m}(x)$ to estimate a vector of the numbers of ones in the buckets of registers after application of the layers containing an (ε, m)-block. We assume that the sequence of layers of the considered network is of the form $XULY$, where U and L are the upper and lower parts of the (ε, m)-block respectively, and X and Y are arbitrary sequences of standard layers. We consider the configurations obtained after the whole iterations and after the XU parts of the iterations.

Lemma 3.20 *Let $R = \{1, \ldots, lm\}$ be a set of registers. Let U and L be respectively the upper and the lower part of an (ε, m)-block on R. Let X and Y be any sequences of standard layers over R. Let c be any configuration of R over $\{0, 1\}$, and let $|\{i \mid c(i) = 1\}| = km + m'$, where k is an integer and $0 \leq m' < m$. For $t \geq 0$ let the sequence of configurations c_t be defined as follows:*

- $c_0 = c$,
- $c_t = XU(c_{t-1})$ *for odd* $t \geq 1$,

- $c_t = LY(c_{t-1})$ *for even* $t \geq 2$.

For each $t \geq 0$, *for* $1 \leq i \leq l$, *let* $y_{t,i} = |\{j \in K_i \mid c(j) = 1\}|$, *where* K_i *is the* i*th* m*-bucket, and let* $y_t = (y_{t,1}, \ldots, y_{t,l})$. *Let* $x = (x_1, \ldots, x_l)$ *be a vector such that, for* $1 \leq i \leq k$, $x_i = m$ *and* $x_{k+1} = m'$ *and for* $k+1 < i \leq m$, $x_i = 0$.

Then for each $t \geq 0$, $\mathcal{V}^t_{\varepsilon,m}(x) \preceq y_t$.

Proof. The vector x is the minimal element in $[0,m]^l$ in relation $\preceq$ such that hd_l equals $km + m'$. Thus $x \preceq y_0$. It follows by induction from Lemmas 3.14 and 3.18 that $\mathcal{V}^t_{\varepsilon,m}(x) \preceq y_t$. □

The proof of the following combinatorial lemma (included in Appendix A) has been invented by Grzegorz Stachowiak. Here we consider only the values of $\mathrm{hd}_i(\mathcal{V}^t_{\varepsilon,m}(x))$ and $\mathrm{tl}_{m,t}(\mathcal{V}^t_{\varepsilon,m}(x))$, where the vector x is of the form $(m)^k(0)^{l-k}$ (i.e. a minimal vector with the sum of coordinates equal to km). The lemma states that, for $t = \alpha l$, the sum of the coordinates of the vector $\mathcal{V}^t_{\varepsilon,m}(x)$ that are outside the last $k + \beta \log(lm)$ coordinates is less than one, where α and β are constant. That is, almost all the weight of the vector is shifted to the last $k + \beta \log(lm)$ coordinates. Such a vector corresponds to an "almost" sorted configuration of zeroes and ones. The lemma also states analogous result for the tail of the $\mathcal{V}^t_{\varepsilon,m}(x)$.

Lemma 3.21 *Let* $0 \leq \varepsilon < \frac{1}{3}$. *There exist constants* α, β *such that, for each* $m > 0$, *for each positive integers* l *and* k, *such that* $ml \geq 12$ *and* $k \leq l$, *for each vector* $x = (m)^k(0)^{l-k}$ *the vector* $y = \mathcal{V}^{\alpha l}_{\varepsilon,m}(x)$ *has following properties:*

- $\mathrm{hd}_{\lfloor l-k-\beta \log(lm) \rfloor}(y) < 1$, *and*
- $\mathrm{tl}_{m,\lceil l-k+\beta \log(lm) \rceil}(y) < 1$.

Proof. See Appendix A. □

Note that $\mathrm{hd}_{\lfloor l-k-\beta \log(lm) \rfloor}(y)$ can be used to upper bound the number of ones in the buckets K_1 through $K_{\lfloor l-k-\beta \log(lm) \rfloor}$. Since the last number is non-negative integer, it must be zero if $\mathrm{hd}_{\lfloor l-k-\beta \log(lm) \rfloor}(y) < 1$. For this reason estimations of the form $\mathrm{hd}_{\lfloor l-k-\beta \log(lm) \rfloor}(y) < 1$ and $\mathrm{tl}_{m,\lceil l-k+\beta \log(lm) \rceil}(y) < 1$ are all we need.

In Lemma 3.21 we assume that the number of ones is a multiple of m. This can be easily generalized to the case where the number of ones is arbitrary:

Corollary 3.22 *Let* ε, α, β, m, *and* l *be as in Lemma 3.21. Let* k *be a non-negative integer,* $k < l$. *Let* $x = (m)^k(m')(0)^{l-k-1}$, *where* $0 \leq m' \leq m$. *Then vector* $y = \mathcal{V}^{\alpha l}_{\varepsilon,m}(x)$ *has the following properties:*

- $\mathrm{hd}_{\lfloor l-k-1-\beta \log(lm) \rfloor}(y) < 1$, *and*
- $\mathrm{tl}_{m,\lceil l-k+\beta \log(lm) \rceil}(y) < 1$.

Proof. The corollary follows from Lemma 3.21 and from the fact that for all $t \geq 0$, for $1 \leq i \leq l$,

$$\mathrm{hd}_i(\mathcal{V}^t_{\varepsilon,m}(x)) \leq \mathrm{hd}_i(\mathcal{V}^t_{\varepsilon,m}((m)^{k+1}(0)^{l-k-1})) \tag{1}$$

and

$$\mathrm{tl}_{m,i}(\mathcal{V}^t_{\varepsilon,m}((m)^k(0)^{l-k})) \geq \mathrm{tl}_{m,i}(\mathcal{V}^t_{\varepsilon,m}(x)).$$

The first inequality (1) follows by induction on t from Lemma 3.16. The second inequality can be shown in a similar way. □

Lemma 3.23 *Let $0 < \varepsilon < \frac{1}{3}$. Let $R = \{1, \ldots, n\}$ be a sequence of registers. Let $M = CN(n, d, R, L)$ be an (ε, m)-block. Let c be a mp-dirty configuration of R, with p such that $m(p+1) \geq 12$. Let X and Y be two sequences of standard layers. There exist two positive values α' and β' that depend only on ε such that the configuration $c' = (XLY)^{\lceil \alpha' p \rceil}(c)$ is $m\lceil \beta' \log((p+1)m) \rceil$-dirty.*

Proof. Let K_j be the first bucket of M that contains a one. Then K_{j+p} is the last bucket that may contain a zero. Let $S = \bigcup_{j \leq i \leq j+p} K_i$. Let L' (respectively, X' and Y') be a S-restriction of L (respectively, of X and Y). All comparators that are in XLY are standard comparators, thus all comparators that are not in $X'L'Y'$ do not change the values in their registers and the S-restriction of c' is equal to $(X'L'Y')^{\lceil \alpha p \rceil}(c_S)$, where c_S is the S-restriction of c. Note that L' is a sequence of layers of an (ε, m)-block on the registers of S. Let L'_1 be the upper part of L' and L'_2 be the lower part of L'.

Let $c_0 = c_S$ and for odd $t \geq 1$, let $c_t = X'L'_1(c_{t-1})$ and let $c_{t+1} = L'_2Y'(c_t)$. For $t \geq 0$, let $x_t = (x_{t,1}, \ldots, x_{t,p+1})$ be a vector such that $x_{t,i}$ is the number of ones in K_{j+i-1} in configuration c_t. Let $q = \sum_{i=1}^{l} x_{0,i}$ (i.e. q is the number of ones in c_S). Let $k = \lfloor q/m \rfloor$ and $m' = q - km$. We define $x' = (m)^k(m')(0)^{l-k-1}$ and for $t \geq 0$ let $x'_t = \mathcal{V}^t_{\varepsilon,m}(x')$. That is, x' is the smallest vector with respect to $\preceq$ that represents a configuration with the same number of ones as c_S.

It follows from Lemma 3.20 that for each $t \geq 0$, $x'_t \preceq x_t$. That means that for each i, $1 \leq i \leq l$,

$$\mathrm{hd}_i(x'_t) \geq \mathrm{hd}_i(x_t)$$

and

$$\mathrm{tl}_{m,i}(x'_t) \geq \mathrm{tl}_{m,i}(x_t).$$

By Corollary 3.22, there exist two constants α and β, such that

$$\mathrm{hd}_{\lfloor p-k-\beta \log((p+1)m) \rfloor}(x'_{\lceil \alpha p \rceil}) < 1$$

and

$$\mathrm{tl}_{m,\lceil p+1-k+\beta \log((p+1)m) \rceil}(x'_{\lceil \alpha p \rceil}) < 1.$$

Thus

$$\mathrm{hd}_{\lfloor p-k-\beta \log((p+1)m) \rfloor}(x_{\lceil \alpha p \rceil}) = 0$$

and

$$\mathrm{tl}_{m,\lceil p+1-k+\beta \log((p+1)m) \rceil}(x_{\lceil \alpha p \rceil}) = 0,$$

since they must be nonnegative integers. Note that

$$\lceil p+1-k+\beta \log((p+1)m) \rceil - \lfloor p-k-\beta \log((p+1)m) \rfloor \leq 2 + 2\beta \log((p+1)m).$$

Thus we can choose the constants α' and β' as respectively α and $2\beta + 2$. □

3.3 Periodic sorting network definition and analysis

The structure of our network is following: The layers are divided into $k+1$ groups, each of them corresponding to an (ε, m_i)-block, for carefully chosen sizes m_i. The idea is that the computation can be divided into virtual phases: During Phase 1, we shrink the size of a dirty region to from $n = m_1 n^{1/k}$ to $m_2 n^{1/k}$. For this purpose we use (ε, m_1)-block as in Lemma 3.23 and disregard in the analysis other layers. We only note that they are standard layers. Then we start the second virtual phase. For this purpose we consider only the buckets of the (ε, m_2)-block that intersect the dirty region. Again we use Lemma 3.23 to show that the size of dirty region is shrunk to $m_3 n^{1/k}$. We iterate this approach k times until we get a sequence that is $O(\log^{k+1} n)$-dirty. Then we apply odd-even transposition sorting network as the last block. This allows to finish sorting in the time proportional to the size of the last dirty region.

Definition 3.24 *Let $0 < \varepsilon < \frac{1}{3}$. Let β' be the constant defined in Lemma 3.23. Let $k \geq 2$ be a positive integer. For a positive integer $n \geq 2^{k+2}$, we define a network $I_{\varepsilon,k,n}$ as a comparator network $CN(n, 2kd_\varepsilon + 2, \{1, \ldots, n\}, L)$, where $L = L_1 L_2 \ldots L_k L_{k+1}$, such that for each i, $1 \leq i \leq k$, L_i is a sequence of layers of (ε, m_i)-block on $\{1, \ldots, n\}$, where*

- $m_1 = \lceil n^{(k-1)/k} \rceil$, *and*
- *for* $2 \leq i \leq k$, $m_i = \lceil m_{i-1}/n^{1/k} \rceil \lceil \beta' \log(2n) \rceil$,

and L_{k+1} is a sequence of the two layers of n-odd-even transposition network.

Note that for $k \geq 2$ and $n \geq 2^{k+2}$ the following holds: $n^{1/k} \geq 2$ and $\lceil n^{(k-1)/k} \rceil (n^{1/k} + 1) \leq 2n$. (Indeed: $(n^{(k-1)/k} + 1)(n^{1/k} + 1) \leq 2n$ if and only if $(n^{(k-1)/k} - 1)(n^{1/k} - 1) \geq 2$. On the other hand $n^{1/k} - 1 \geq 2^{(k+2)/k} - 1 > 1$ and $n^{(k-1)/k} - 1 \geq 2^{(k+2)(k-1)/k} - 1 \geq 3$ since $k \geq 2$.)

We assume that n is large enough, to have $m_i n^{1/k} \geq 12$, for $1 \leq i \leq k$.

Theorem 3.25 *The network $I_{\varepsilon,k,n}$ sorts any input in $O(kn^{1/k})$ iterations.*

Proof. Let α' and β' be the constants α' and β' from Lemma 3.23. Note that for each i, $1 \leq i \leq k$, the sequence of layers of $I_{\varepsilon,k,n}$ has a following structure: $L = X_i L_i Y_i$, where L_i is a sequence of layers of the (ε, m_i)-block, and X_i and Y_i are sequences of standard layers.

Claim 3.26 *Let c_0 be any configuration over $\{0,1\}$ and, for $0 < i < k$, let $c_i = (X_i L_i Y_i)^{\lceil \alpha' n^{1/k} \rceil}(c_{i-1})$. Then, for $0 \leq i < k$, c_i is $(m_{i+1} n^{1/k})$-dirty.*

Obviously, c_0 is at most $(m_1 n^{1/k})$-dirty. By Lemma 3.23 if c_{i-1} is $(m_i n^{1/k})$-dirty, then c_i is at most $(m_i \lceil \beta' \log((n^{1/k} + 1)m_i) \rceil)$-dirty. But

$$m_i \lceil \beta' \log((n^{1/k} + 1)m_i) \rceil \leq n^{1/k} \lceil m_i / n^{1/k} \rceil \lceil \beta' \log(2n) \rceil = n^{1/k} m_{i+1}.$$

It follows that $c_k = L^{k \lceil \alpha' n^{1/k} \rceil}(c_0)$ is $(m_k \lceil \beta' \log((n^{1/k} + 1)m_k) \rceil)$-dirty.

Claim 3.27 *For* $1 \le i \le k$,

$$m_i \le \left(n^{(k-i)/k} + \sum_{j=0}^{i-1} 1/n^{j/k} \right) \lceil \beta' \log(2n) \rceil^{i-1}.$$

It follows from the definition of m_i that

$$m_1 \le n^{(k-1)/k} + 1$$

and that, for $1 \le i \le k-1$, if

$$m_i \le \left(n^{(k-i)/k} + \sum_{j=0}^{i-1} 1/n^{j/k} \right) \lceil \beta' \log(2n) \rceil^{i-1},$$

then

$$m_{i+1} \le \left(n^{(k-i-1)/k} + \sum_{j=0}^{i} 1/n^{j/k} \right) \lceil \beta' \log(2n) \rceil^{i}.$$

Indeed:

$$\begin{aligned} m_{i+1} &= \lceil m_i / n^{1/k} \rceil \lceil \beta' \log(2n) \rceil \\ &\le \left\lceil \left(n^{(k-i)/k} + \sum_{j=0}^{i-1} 1/n^{j/k} \right) / n^{1/k} \cdot \lceil \beta' \log(2n) \rceil^{i-1} \right\rceil \lceil \beta' \log(2n) \rceil \\ &\le \left\lceil \left(n^{(k-i)/k} + \sum_{j=0}^{i-1} 1/n^{j/k} \right) / n^{1/k} \right\rceil \cdot \lceil \beta' \log(2n) \rceil^{i} \\ &\le \left(\left(n^{(k-i)/k} + \sum_{j=0}^{i-1} 1/n^{j/k} \right) / n^{1/k} + 1 \right) \cdot \lceil \beta' \log(2n) \rceil^{i} \\ &= \left(n^{(k-i-1)/k} + \sum_{j=0}^{i} 1/n^{j/k} \right) \cdot \lceil \beta' \log(2n) \rceil^{i}. \end{aligned}$$

By Claim 3.27, $m_k \le \left(1 + \sum_{j=0}^{k} 1/n^{j/k}\right) \lceil \beta' \log(2n) \rceil^{k}$.

We assume that $n^{1/k} \ge 2$, so $m_k \le 3\lceil \beta' \log(2n) \rceil^{k}$ and $m_k \lceil \beta' \log((n^{1/k} + 1)m_k) \rceil \le 3\lceil \beta' \log(2n) \rceil^{k+1}$. Thus c_k is $(3\lceil \beta' \log(2n) \rceil^{k+1})$-dirty. $L = XL_{k+1}$, where X is a sequence of standard layers and L_{k+1} is the sequence of the two layers of n-odd-even transposition network. By Lemma 3.4 such a network sorts the $(3\lceil \beta' \log(2n) \rceil^{k+1})$-dirty configuration in $3\lceil \beta' \log(2n) \rceil^{k+1}$ iterations. Thus the total number of the iterations of $I_{\varepsilon,k,n}$ needed for sorting arbitrary n-dirty configuration c_0 is

$$k\lceil \alpha' n^{1/k} \rceil + 3\lceil \beta' \log(2n) \rceil^{k+1} = O(kn^{1/k}).$$

The sorting time (i.e. the depth of $I_{\varepsilon,k,n}$ multiplied by the number of iterations) is

$$T_{n,k} = (2kd_\varepsilon + 2)(k\lceil \alpha' n^{1/k} \rceil + 3\lceil \beta' \log(2n) \rceil^{k+1}),$$

where d_ε is the depth of the ε-halver. Since d_ε is constant, we have

$$T_{n,k} = O(k^2 n^{1/k}).$$

□

4 Correction Network

In this section we consider a problem of sorting sequences of length n that are obtained from a sorted sequence by changing the values of at most its k elements, where k is much smaller than n.

Definition 4.1 *A sequence $(a_1, \ldots, a_n)$ is called* k-disturbed *if and only if it can be obtained from some sorted sequence s by changing values of at most k elements of s. A configuration c of the set of n registers R is called a* k-disturbed configuration *if and only if a sequence $(c(R_1), \ldots, c(R_n))$ is k-disturbed.*

The expression "k-disturbed" should be understood "at most k-disturbed". Note that sequence is k-disturbed if and only if it can be transformed into a sorted sequence by changing at most k of its elements. Note also that a 0-disturbed sequence is a sorted sequence.

The main result presented in this section is the following theorem:

Theorem 4.2 *Let n and k be arbitrary positive integers such that $k \leq n$. Then there is an explicit construction of a comparator network of depth $4 \log n + O(\log^2 k \log \log n)$ that sorts any k-disturbed input sequence.*

Note for $k = o\left(2^{\sqrt{\log n / \log \log n}}\right)$ the depth of the network is $4 \log n + o(\log n)$.

4.1 Correction networks. Preliminaries

Definition 4.3 *Let R be a set of n registers. A network $N = CN(n, d, R, L)$ is called a* k-correction network *on R if and only if for each k-disturbed configuration c of R, the sequence $(L(d, c)(R_1), \ldots, L(d, c)(R_n))$ is sorted.*

Definition 4.4 *Let R be a set of n registers. Let c be any configuration of R over $\{0, 1\}$ such that $|\{i \mid c(R_i) = 0\}| = x$. Then the* zeroes area of c *(respectively* ones area of c*) denotes the set of registers $\{R_i \mid 1 \leq i \leq x\}$ (respectively $\{R_i \mid x + 1 \leq i \leq n\}$). We call a* displaced one *a one contained in a register from the zeros area. We call a* displaced zero *a zero contained in a register from the ones area.*

Lemma 4.5 *Let R be a set of n registers. Let $k \geq 0$. Let c be any k-disturbed configuration of R over $\{0, 1\}$. Then c has at most k displaced zeroes and at most k displaced ones.*

Proof. Suppose that c has more than k zeroes in the ones area. Then there has to be more than k ones in the zeroes area. Then in the sequence $s = (c(R_1), \ldots, c(R_n))$ a group of at least $k + 1$ ones is entirely on the left side of a group of at least $k + 1$ zeroes. To change the sequence s into a sorted sequence we have to change the values of at least $k + 1$ elements. Thus c is not k-disturbed. □

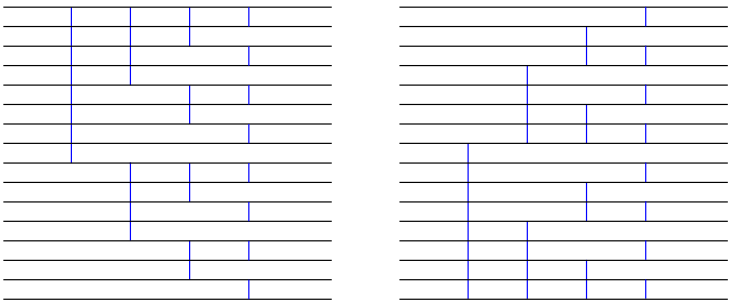


Figure 7: The networks INS^1_{16} and INS^0_{16}

Lemma 4.6 *Let R be a set of n registers. Let c be any k-disturbed configuration of R over X. For each l, $1 \leq l \leq n$ let c_l be the l-threshold of c. Then c_l is k-disturbed.*

Proof. Suppose that for some l, c_l is not k-disturbed. Let

$$x_l = \min\{c(R_i) \mid c_l(R_i) = 1\}.$$

Let c' be a configuration obtained from c by changing at most k of its values such that the sequence $(c'(R_1), \ldots, c'(R_n))$ is sorted. Let c'' be a configuration of R over $\{0,1\}$ such that $c''(R_i) = 0$ if and only if $c'(R_i) < x_l$. Then c'' can be obtained from c_l by changing at most k of its values, since $c_l(R) \neq c''(R)$ if and only if $c(R) \geq x_l$ and $c'(R) < x_l$ or $c(R) > x_l$ and $c'(R) \leq x_l$ (i.e. R is one of the at most k registers, where c and c' differ). On the other hand, the sequence $(c''(R_1), \ldots, c''(R_n))$ is sorted. Contradiction with the fact that c_l is not k-disturbed. □

Lemma 4.7 *The comparator network N is a k-correction network if and only if N sorts all k-disturbed zero-one sequences.*

Proof. It follows from Lemmas 4.6 and 2.22. □

4.2 Auxiliary networks

The following definitions introduce the classical insertion networks INS^1_n and INS^0_n. The network INS^1_n inserts any value placed in its first register to the sorted sequence stored in the remaining registers. (That is, INS^1_n sorts any sequence that differs from the sorted sequence only at the first position.) Analogously, the network INS^0_n sorts any sequence that differs from the sorted sequence only at the last position.

Definition 4.8 *Let m be a positive integer. For $n = 2^m$, we define a network $INS_n^1 = CN(n, m, R, L)$ on the set of registers $R = \{1, \ldots, n\}$ with the following layers $L = (L_1, \ldots, L_m)$ (see Fig. 8):*

- *if $m = 1$, then $L = (\{(1, 2)\})$,*
- *if $m > 1$, then $L = L'(L_m)$, where:*
  - *$L_m = \{(i, i+1) \mid 1 \leq i \leq n-1, \quad i \text{ is odd}\}$*
  - *L' is the f-mapping of the sequence of layers of $INS_{n/2}^1$, where $f(x) = 2x - 1$.*

If $n > 1$ is not a power of two, then

$$INS_n^1 = CN(n, \lceil \log n \rceil, \{1, \ldots, n\}, L''),$$

where L'' is a $\{1, \ldots, n\}$-restriction of the sequence of layers of $INS_{2^{\lceil \log n \rceil}}^1$.

Definition 4.9 *The network INS_n^0 is dual to INS_n^1. That is*

$$INS_n^0 = CN(n, \lceil \log n \rceil, \{1, \ldots, n\}, L),$$

where $L = (L_1, \ldots, L_{2\lceil \log n \rceil - 1})$, and $L_i = \{(n - j + 1, n - i + 1) \mid (i, j) \in L_i'\}$, where L_i' denotes the ith layer of INS_n^1.

Below we define simple networks I_n^1 and I_n^0 that are able to sort 1-disturbed sequence of zeroes and ones provided that a zero has been changed to a one (in the case of I_n^1) or a one has been changed to zero (in the case of I_n^0). Examples of these networks are depicted on Fig. 8.

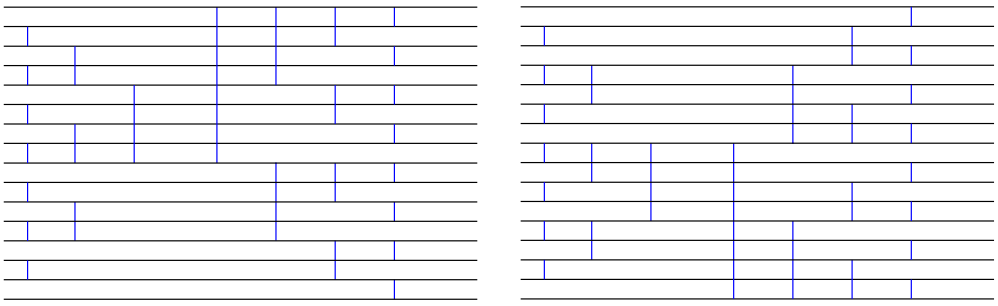
Definition 4.10 *Let m be a positive integer. For $n = 2^m$, we define a network $I_n^1 = CN(n, 2m - 1, R, L)$ on the set of registers $R = \{1, \ldots, n\}$ with the following layers $L = (L_1, \ldots, L_{2m-1})$:*

- *if $m = 1$, then $L = (\{(1, 2)\})$,*
- *if $m > 1$, then $L = (L_1)L'(L_{2m-1})$, where:*
  - *$L_1 = \{(i, i+1) \mid 2 \leq i \leq n-2, \quad i \text{ is even}\}$*
  - *$L_{2m-1} = \{(i, i+1) \mid 1 \leq i \leq n-1, \quad i \text{ is odd}\}$*
  - *L' is the f-mapping of the sequence of layers of $I_{n/2}^1$, where $f(x) = 2x - 1$.*

If $n > 1$ is not a power of two, then

$$I_n^1 = CN(n, 2\lceil \log n \rceil - 1, \{1, \ldots, n\}, L''),$$

where L'' is a $\{1, \ldots, n\}$-restriction of the sequence of layers of $I_{2^{\lceil \log n \rceil}}^1$.



I^{1}_{16} I^{0}_{16}

Figure 8: The networks I^1_{16} and I^0_{16}

Definition 4.11 *The network I^0_n is dual to I^1_n. That is*

$$I^0_n = CN(n, 2\lceil \log n \rceil - 1, \{1, \ldots, n\}, L),$$

where $L = (L_1, \ldots, L_{2\lceil \log n \rceil - 1})$, and $L_i = \{(n - j + 1, n - i + 1) \mid (i, j) \in L'_i\}$, where L'_i denotes the ith layer of I^1_n.

Lemma 4.12 *The network I^1_n (respectively I^0_n) sorts any zero-one input sequence that has been obtained from a sorted zero-one sequence by changing a single zero into a one (respectively, a single one into a zero).*

Proof. We prove only the lemma for I^1_n. The proof for I^0_n is analogous. For $n = 2$, the lemma is obviously true. Let $n = 2^m$, for some $m > 1$. Let $a = (a_1, \ldots, a_n)$ be a zero-one sequence obtained from a sorted sequence by changing a single zero element into a one. Let $a' = (a'_1, \ldots, a'_n)$ be a sequence that is a result of applying the first layer of I^1_n to the sequence a. Then the subsequence b of a' on the even registers is sorted. The subsequence b' of a' on the odd registers can be obtained from a sorted sequence by changing a single zero into a one. The number of ones in b' is not less than the number of ones in b and not greater than the number of ones in b plus one. Let $c = (c_1, \ldots, c_n)$ be a sequence obtained by applying the next $2m - 3$ layers of I^1_n to a'. The subsequence d of c on even registers is equal to b, since these layers contain no comparators with even registers. The subsequence d' of c on odd registers is a sorted sequence b', since these layers are the mapping of the layers of $I^1_{n/2}$ on the odd registers. If the number of ones in d' is equal to the number of ones in d, then c is already sorted. Otherwise the number of ones in d' is at most one more than the number of ones in d. The last layer of I^1_n shifts the first one in c into next even register and the output becomes sorted.

Note that the network $I^1_{2^{\lceil \log n \rceil}}$ sorts all the zero-one sequences obtained from a sorted zero-one sequence by changing a single zero into a one that have only

ones in registers greater than n. Thus for n that is not a power of two, the lemma follows from Corollary 2.14. □

In the following definitions we introduce a notion of a k-merge version of a comparator network M: a comparator network obtained from M by replacing the registers of M by the buckets of k registers and by replacing the comparators of M by the merging subnetworks on the corresponding pairs of buckets.

Let $BM_k = CN(2k, m_k, \{1, \ldots, 2k\}, B)$ denote the Batcher merging network for two sorted sequences placed in the registers $\{1, \ldots, k\}$ and $\{k+1, \ldots, 2k\}$. Let $BS_k = CN(k, d_k, \{1, \ldots, k\}, B')$ denote the Batcher sorting network for the sequences of length k.

Definition 4.13 *Let $k > 0$. For $i \neq j$, let*

$$f_{i,j} : \{1, \ldots, 2k\} \to \{(i-1)k+1, \ldots, ik\} \cup \{(j-1)k+1, \ldots, jk\}$$

be a bijection defined as follows:

$$f_{i,j}(x) = \begin{cases} (i-1)k + x & \text{if } x \leq k, \\ (j-1)k + (x-k) & \text{if } x > k. \end{cases}$$

Let $M = CN(n, d, R, L)$, where $R = \{1, \ldots, n\}$ and $L = (L_1, \ldots, L_d)$. We call a network $M_k = CN(kn, m_k d, R', L')$ a k-merge version of M *if and only if $R' = \{1, \ldots, kn\}$, and $L' = L'_1 \ldots L'_d$, where for each t, $1 \leq t \leq d$, the subsequence of layers L'_t is the union of $f_{i,j}$-mappings of B for all $(i,j) \in L_t$.*

Let B'' be a union of f_i-mappings of B', where $1 \leq i \leq n$ and $f_i(x) = (i-1)k + x$. Let $M'_k = CN(kn, d_k + m_k d, R', B''L')$. We call M'_k an extended k-merge version of M.

For $1 \leq i \leq n$ we call a subset of registers $K_i = \{r \mid (i-1)k + 1 \leq i \leq ik\}$ the ith bucket *of M_k.*

Lemma 4.14 *Let $S_n = CN(n, d, \{1, \ldots, n\}, L)$ be a 1-correction network of depth d for the input sequences of length n. Let $S_{n,k}$ be the extended k merge version of S_n. Then the $S_{n,k}$ is a k-correction network for the input sequences of length kn.*

Proof. Let $a = (a_1, \ldots, a_{nk})$ be a k-disturbed 0-1 sequence. Let x denote the number of zeroes in a. (We assume that $x > 0$.) Let $a' = (a'_1, \ldots, a'_{nk})$ be a sequence obtained after sorting the buckets within the first d_k layers of $S_{n,k}$ (where d_k is the depth of the Batcher sorting network BS_k used in construction of the $S_{n,k}$). Let $x' = \lceil x/k \rceil$. Thus bucket x' is the last one that intersects the zero area.

We show that after application of the remaining part of $S_{n,k}$ to the sequence $(a'_1, \ldots, a'_{nk})$ all the buckets with indices greater than x' will be cleared from zeroes. Analogous reasoning can be used to show that all the buckets with the numbers less than x' will be cleared from ones. Since all the buckets remain sorted this implies that the whole output is also sorted.

For $1 \leq v \leq y < w \leq n$, let $\gamma_{v,w,y}$ denote a sequence obtained from the sorted 0-1 sequence with exactly y zeroes by changing a zero on position v into

a one and a one on position w into a zero. Let $d_{v,w,y}$ denote the minimal d such that after applying the first d layers of S_n on input $\gamma_{v,w,y}$ we get a sorted sequence. Note that the layer $d_{v,w,y}$ is the first and the only layer within which the displaced zero is compared with the displaced one.

We consider all displaced zeroes in a' in buckets $x'+1,\ldots,n$. We show that $S_{n,k}$ gets rid of displaced zeroes in these buckets. In the same way, we may show that $S_{n,k}$ gets rid of displaced ones in buckets $1,\ldots,x'-1$. Since $S_{n,k}$ outputs bucket x' in a sorted state, it follows that the whole output is sorted.

Let m be the number of zeroes in buckets $x'+1,\ldots,n$ in a' and let W denote the set of their positions. All these zeroes are, of course, displaced. Let l denote the number of ones in buckets 1 through x' in a' and let V be the set of their positions (some of these ones are displaced, those from bucket x' are not necessarily displaced). Obviously, $m \leq l$ and $m \leq k$. For each $j \in W$, we choose an $i \in V$ using an inductive procedure based on the following conditions:

- We set $V_0 = V$ and $W_0 = W$.
- For each t, $1 \leq t \leq m$,
  - we set $i_t = v$ and $j_t = w$, where is (v,w) is one of the pairs from $V_{t-1} \times W_{t-1}$ that minimizes the value $d_{\lceil v/k \rceil, \lceil w/k \rceil, x'}$, and
  - we set $V_t = V_{t-1} \setminus \{i_t\}$ and $W_t = W_{t-1} \setminus \{j_t\}$.

The idea is the following. A displaced zero terminates to be displaced at the moment when the bucket containing it is merged with a bucket with an index at most x' and containing a one. In fact, if the second bucket contains less ones than there are zeroes in the first bucket, then some of the zeroes must remain in the first bucket and are still displaced. For any displaced zero, our definition fixes a one that may cause the zero to finish its status of an displaced element.

Let $\gamma_{v,w,y,t}$ denote the sequence stored in the registers of S_n after applying the first t layers of S_n on input $\gamma_{v,w,y}$. For $1 \leq i \leq n$ and $t \geq 0$, let $p_{i,t}$ denote the number of sequences among $\gamma_{\lceil i_1/k \rceil, \lceil j_1/k \rceil, x', t}, \ldots, \gamma_{\lceil i_m/k \rceil, \lceil j_m/k \rceil, x', t}$ that contain ones at position i. Let $p'_{i,t}$ denote the number of ones in bucket i after applying the first $d_k + t \cdot c_k$ layers of $S_{n,k}$ to input a (where c_k is a depth of a Batcher merging network BM_k used in construction of $S_{n,k}$). We prove the following technical claim:

Claim 4.15 *1. If $1 \leq i \leq x'$, then $p_{i,t} \leq p'_{i,t}$.*
That is, the number of ones in the bucket i at moment t is at least $p_{i,t}$.

2. If $x' < i \leq n$, then $m - p_{i,t} \geq k - p'_{i,t}$.
That is, the number of zeroes in the bucket i at moment t is at most $m - p_{i,t}$.

Proof of the claim. The proof is by induction on t.

For $t = 0$ the properties follow from the way we have defined the sequences $i_1,\ldots,i_m$ and $j_1,\ldots,j_m$. (Property 1 is implied by $\{i_1,\ldots,i_m\} \subseteq V$ and Property 2 is implied by $W \subseteq \{j_1,\ldots,j_m\}$.)

Let $t > 0$. For each register R_i of S_n, $1 \leq i \leq n$, there are three possibilities:

Case 1. There is no comparator incident to R_i in layer t of S_n.

Case 2. There is an comparator (R_j, R_i) in layer t of S_n.

Case 3. There is an comparator (R_i, R_j) in layer t of S_n.

The first case is trivial. We have $p_{i,t} = p_{i,t-1}$ and $p'_{i,t} = p'_{i,t-1}$, so Properties 1 and 2 of the claim follow from the induction hypothesis.

Proof of Property 1 of Claim 4.15. Let $1 \leq i \leq x'$.

In the second case,

$$p_{i,t} = p_{i,t-1} + p_{j,t-1}$$

and, as always, $p_{i,t} \leq m \leq k$. In $S_{n,k}$ there is a network merging buckets j and i in the corresponding layers. Thus

$$p'_{i,t} = \min\{k, p'_{i,t-1} + p'_{j,t-1}\}$$

(since if we merge two buckets containing initially a and b ones, the one with a bigger index will contain $\min\{k, a+b\}$ ones). Combining this with the induction hypothesis we get $p_{i,t} \leq p'_{i,t}$.

In the third case, there are two sub-cases: either $j \leq x'$ or $j > x'$. If $j \leq x'$, then $p_{i,t} = 0$ and hence $p_{i,t} \leq p'_{i,t}$. The reason is that a one in each of the sequences $\gamma_{\lceil i_q/k \rceil, \lceil j_q/k \rceil, x', t-1}$ can freely move to any position j, $i < j \leq x'$.

The sub-case $j > x'$ is more tedious. We claim that

$$p_{i,t} \leq \max\{0, p_{i,t-1} - (m - p_{j,t-1})\}.$$

Indeed, if $\gamma_{\lceil i_r/k \rceil, \lceil j_r/k \rceil, x', t-1}$ contains a displaced one at position i and a displaced zero at position j, then $\gamma_{\lceil i_r/k \rceil, \lceil j_r/k \rceil, x', t}$ contains a zero at position i. Therefore it contributes to the decrease of p_i. So if $p_{i,t} > \max\{0, p_{i,t-1} - (m - p_{j,t-1})\}$, then there are two different pairs (i_r, j_r) and $(i_{r'}, j_{r'})$ such that $\gamma_{\lceil i_r/k \rceil, \lceil j_r/k \rceil, x', t}$ contains a displaced one at position i and $\gamma_{\lceil i_{r'}/k \rceil, \lceil j_{r'}/k \rceil, x', t}$ contains a displaced zero at position j. Then of course, $d_{\lceil i_r/k \rceil, \lceil j_r/k \rceil, x'} > t$ and $d_{\lceil i_{r'}/k \rceil, \lceil j_{r'}/k \rceil, x'} > t$, since we have detected displaced elements after step t on positions, respectively, i and j. On the other hand, $d_{\lceil i_r/k \rceil, \lceil j_{r'}/k \rceil, x'} \leq t$, since in the worst case the displaced zero and displaced one meet at layer t. So we should have chosen a pair $(i_r, j_{r'})$ instead of the first of (i_r, j_r) and $(i_{r'}, j_{r'})$.

On the other hand,

$$p'_{i,t} = \max\{0, p'_{i,t-1} - (k - p'_{j,t-1})\}.$$

By the induction hypothesis, $p_{i,t-1} \leq p'_{i,t-1}$ and $(m - p_{j,t-1}) \geq (k - p'_{j,t-1})$. Combining all this we get $p_{i,t} \leq p'_{i,t}$.

Proof of Property 2 of Claim 4.15. Let $x'+1 \leq i \leq n$.

In the second case, there are two sub-cases possible: either $j > x'$ or $j \leq x'$. In the first sub-case $k - p'_{i,t} = 0$, since the total number of zeroes in buckets j and i is not greater than m, $m \leq k$, and the corresponding merging sub-network moves all the zeroes to the jth bucket. Hence Property 2 holds.

Now consider the second sub-case. Note that

$$m - p_{i,t} \geq (m - p_{i,t-1}) - p_{j,t-1},$$

since in at most $p_{j,t-1}$ cases $\gamma_{\lceil i_r/k \rceil, \lceil j_r/k \rceil, x', t-1}$ contains a one on position j. Thus, for at most $p_{j,t-1}$ cases a zero at position i is exchanged with a one at step t. On the other hand,

$$k - p'_{i,t} = \max\{0, (k - p'_{i,t-1}) - p'_{j,t-1}\}.$$

By the induction hypothesis, $k - p'_{i,t-1} \leq m - p_{i,t-1}$ and $p_{j,t-1} \leq p'_{j,t-1}$. Hence $k - p'_{i,t} \leq m - p_{i,t}$.

In the third case

$$m - p_{i,t} = (m - p_{i,t-1}) + (m - p_{j,t-1})$$

and

$$k - p'_{i,t} = \min\{k, (k - p'_{i,t-1}) + (k - p'_{j,t-1})\}.$$

So Claim 4.15 follows by the induction hypothesis.

Since S_n sorts each sequence $\gamma_{\lceil i_1/k \rceil, \lceil j_1/k \rceil, x', t}$ through $\gamma_{\lceil i_m/k \rceil, \lceil j_m/k \rceil, x', t}$, we have $p_{i,t} = m$, for $i > x'$ and for t equal to the depth of S_n. By Claim 4.15(b), $p'_{i,t}$ must be also equal m, for $i > x'$ (i.e. the ith bucket must not contain zeroes). Thus Lemma 4.14 follows from Claim 4.15(b) and its dual version for ones (which we skip here). □

4.3 Construction of correction network $N_{n,k}$

In this section we describe the construction of the k-correction network for k-disturbed sequences of length $n > 256$, where $3 \leq k \leq \frac{1}{2} n^{\frac{1}{3+\log\log n}}$. This network will be denoted by $N_{n,k} = (n, D, \{1, \ldots, n\}, L)$. By R we denote the set of registers of $N_{n,k}$ (i.e. $\{1, \ldots, n\}$).

The sequence L of layers of $N_{n,k}$ is divided into five parts called *phases*. (Thus $L = P_1 P_2 P_3 P_4 P_5$, where P_i denotes the ith phase.) Construction of each phase is described in a separate subsection.

We assume that n is divisible by n_2, where n_2 is defined in the description of Phase 4 of the network. Here we only assume that n_2 is even and $n_2 > 2k$.

First we arrange the n registers of $N_{n,k}$ in a matrix M of size $n_1 \times n_2$, where $n_1 = n/n_2$ (i.e. n_1 is the number of rows and n_2 is the number of columns) in the row-major order. The rows are numbered 1 through n_1 starting at the top of the matrix and the columns are numbered 1 through n_2 starting at the leftmost column. So the ith row (denoted by ROW_i) contains registers $(i-1) \cdot n_2 + 1, \ldots, i \cdot n_2$, the jth column (denoted by COL_j) contains the registers

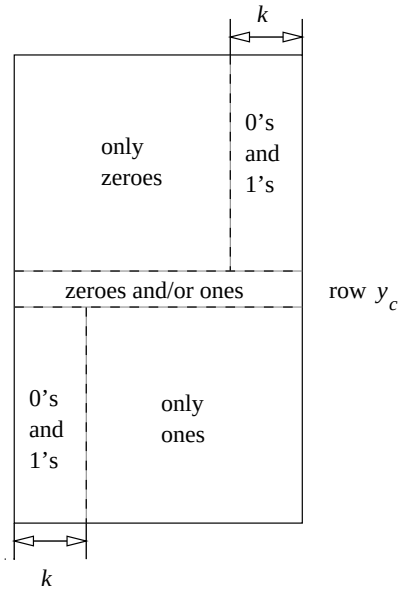


Figure 9: Configuration of zeroes and ones in M after Phase 1

$j + (k-1) \cdot n_2$, for $1 \leq k \leq n_1$. We use a convention that for $i < 1$ or $i > n_1$, $ROW_i = \emptyset$ and for $j < 1$ or $j > n_2$, $COL_i = \emptyset$.

Definition 4.16 *For any zero-one configuration c of R we define y_c as $\lceil x/n_1 \rceil$ where $x = |\{i \mid c(i) = 0\}|$*

Note that the rows $1, \ldots, y_c - 1$, are contained in the zeroes area of c and the rows $y_c + 1, \ldots, n_1$ are contained in the ones area. ROW_{y_c} may intersect both areas.

By S_n we denote the Schimmler and Starke [15] network for input of size n. S_n is a network of depth $2\lceil \log n \rceil - 1$ similar to I_n^0 and I_n^1 that is a complete 1-correction network (i.e. it sorts arbitrary 1-disturbed sequence). By $S_{n,k}$ we denote the extended k-merge version of S_n.

4.3.1 Phase 1

Let L' be a $\{1, \ldots, n_2\}$-restriction of the sequence of layers of $S_{\lceil n_2/k \rceil, k}$. We define layers P_1 as the union of f_i-mappings of ROW_i-restrictions of L', for $1 \leq i \leq n_1$ and $f_i(x) = n_2(i-1) + x$.

Lemma 4.17 *Let c be any k-disturbed configuration of R. Then for each i, $1 \leq i \leq n_1$, the ROW_i-restriction of $P_1(c)$ is sorted.*

Proof. Follows from Lemma 4.14 and from the fact that each ROW_i-restriction of c is also k-disturbed. □

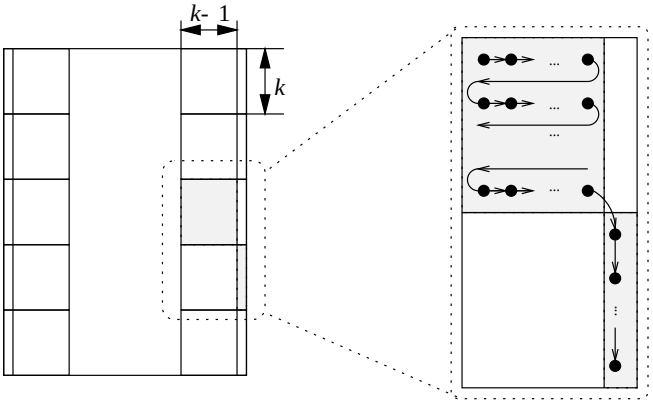


Figure 10: A single right cluster in M and the ordering of registers inside the cluster

Corollary 4.18 *Let c be any k-disturbed configuration of R over $\{0,1\}$. Then the configuration $c' = P_1(c)$ has following properties (see Fig. 9):*

1. *For each i, $1 \leq i < y_c$, the sequence corresponding to ROW_i-restriction of c' is sorted and contains all its ones in the last k positions.*
2. *For each i, $y_c < i \leq n_1$, the sequence corresponding to ROW_i-restriction of c' is sorted and contains all its zeroes in the first k positions.*
3. *The sequence corresponding to ROW_{y_c}-restriction of c' is sorted.*

4.3.2 Phase 2

The aim of Phases 2 and 3 is to move the displaced zeroes that are below the row $y_c + 1$ to the leftmost column and the displaced ones that are above the row $y_c - 1$ to the rightmost column.

We partition the sub-matrix of k rightmost (respectively, leftmost) columns into squares of size $k \times k$. For each square contained in rightmost (respectively, leftmost) columns, the subset consisting of the first $k-1$ columns of the square and the last column of next lower square (respectively first column of the square and the $k-1$ last columns of the next lower square) is called a *cluster* (see Fig. 10).

During Phase 2 each cluster is sorted by a Batcher sorting network for input size k^2.

Let us describe this more formally. First we define the increasing function f_j such that that the set $f_j(\{1,\ldots,k^2\}) \cap R$ is the jth left cluster. If $f_j(\{1,\ldots,k^2\}) \subseteq R$, then the register $f_j(i)$ is the ith register of the jth left cluster according to their natural order.

It is easy to check that f_j is defined as follows for $x \in \{1, \ldots, k^2\}$:

$$f_j(x) = \begin{cases} (j-1)kn_2 + (x-1)n_2 + 1 & \text{if } x \leq k, \\ j(k+i_x-1)n_2 + (x-k) + 1 - (i_x-1)(k-1) & \text{if } x > k, \end{cases}$$

where $i_x = \lceil \frac{x-k}{k-1} \rceil$.

For each j, $0 \leq j \leq \lceil n_1/k \rceil$, let $CL_j = f_j(\{1, \ldots, k^2\}) \cap R$ denote the jth *left cluster*. Note that all the sets CL_j are pairwise disjoint. We call a subset $TL_j = f_j(\{1, \ldots, k\}) \cap R$ a *tail* of the jth left cluster. Note that each TL_j is contained in COL_1 and intersects the rows $(j-1)k+1$ through jk. For each i, $1 \leq i \leq k$, we call a subset $RL_{j,i} = f_j(\{k+(i-1)(k-1)+1, \ldots, k+i(k-1)\}) \cap R$ the ith *row* of the jth left cluster. Note that each $RL_{j,i}$ is contained in ROW_{jk+i} and intersects the columns 2 trough k. If $i < 1$ or $i > k$, then we assume that $RL_{j,i} = \emptyset$.

In a similar way, we define the right clusters with the use of the following increasing functions g_j:

$$g_j(x) = (j+1)kn_2 - f_1(k^2 - x + 1) + 1.$$

For each j, $0 \leq j \leq \lceil n_1/k \rceil$, let $CR_j = g_j(\{1, \ldots, k^2\}) \cap R$ denote the jth *right cluster*. All the sets CR_j are pairwise disjoint.

We call a subset $TR_j = g_j(\{k^2 - k + 1, \ldots, k^2\}) \cap R$ a *tail* of the jth right cluster. Each TR_j is contained in COL_{n_2} and intersects the rows $jk+1$ through $(j+1)k$. For each i, $1 \leq i \leq k$ we call a subset $RR_{j,i} = g_j(\{(i-1)(k-1)+1, \ldots, i(k-1)\}) \cap R$ the ith *row* of the CR_j. Each $RR_{j,i}$ is contained in $ROW_{(j-1)k+i}$ and intersects the columns $n_2 - k$ through $n_2 - 1$. If $i < 1$ or $i > k$, then we assume that $RR_{j,i} = \emptyset$.

Recall that BS_{k^2} denote a Batcher sorting network for the set of registers $\{1, \ldots, k^2\}$. Let L' be a sequence of layers of BS_{k^2}. The sequence of layers P_2 (i.e. of Phase 2) is the R-restriction of the union of f_j-mappings and g_j-mappings of L', for $0 \leq j \leq \lceil n_1/k \rceil$.

Let c be any k-disturbed configuration of R over $\{0,1\}$. The configuration $c' = P_1P_2(c)$ has the following properties:

Fact 4.19 *Let*

$$U = \left(\bigcup_{1 \leq i < y_c} ROW_i \right) \setminus \left(\bigcup_{n_2 - k < j \leq n_2} COL_i \right)$$

and

$$D = \left(\bigcup_{y_c < i \leq n_1} ROW_i \right) \setminus \left(\bigcup_{1 \leq j \leq k} COL_i \right)$$

The U-restriction of c' contains only zeroes and the D-restriction of c' contains only ones.

Proof. Fact 4.19 follows immediately from the Corollary 4.18 and from the fact that each comparator of the layers of P_2 is contained either in the k leftmost or in the k rightmost columns of M. □

Fact 4.20 *Let*

$$U' = \left(\bigcup_{1 \leq i < y_c} ROW_i \right) \setminus COL_{n_2}$$

and

$$D' = \left(\bigcup_{y_c < i \leq n_1} ROW_i \right) \setminus COL_1$$

The U'-restriction of c' contains at most $k-1$ ones and the D'-restriction of c' contains at most $k-1$ zeroes.

Proof. In configuration $P_1(c)$ all the rows of M are sorted. The application of P_2 does not decrease the number of ones in the rightmost column and does not decrease the number of zeroes in the leftmost column. Hence at least one of the ones that are above ROW_{y_c} must remain in the rightmost column and at least one of the zeroes that are below ROW_{y_c} must remain in the leftmost column. □

Fact 4.21 *If CR_j is (entirely) above the row y_c of M, then the CR_j-restriction of c' has all its ones in its tail TR_j.*

Proof. For each cluster CR_j lying entirely above the row y_c the CR_j-restriction of $P_1(c)$ contains at most k ones. The CR_j-restriction of $P_1(c)$ is sorted by P_2 and TR_j contains the last k registers of CR_j. □

Fact 4.22 *If TR_j intersects the row y_c of M, then CR_j-restriction of c' has all its ones in $TR_j \cup RR_{j,k}$.*

Proof. By Fact 4.20 and the fact that $CR_j \setminus TR_j$ is on the left side of COL_{n_2} and above ROW_{y_c}, there are at most $k-1$ ones in $(CR_j \setminus TR_j)$-restriction of c'. Since $(CR_j \setminus TR_j)$-restriction of c' is sorted, all its ones must be in the last row $RR_{j,k}$. □

Fact 4.23 *If $(j-1)k + i = y_c$, for $1 \leq i \leq k$, then the CR_j-restriction of c' contains ones only in $RR_{j,i} \cup RR_{j,i-1}$.*

Proof. Fact 4.23 follows from the fact that there are at most $k-1$ ones in the registers above the row y_c of M in the CR_j-restriction of c' and that either $i = 1$ or there is enough space for them in $RR_{j,i-1}$. □

Facts 4.24,4.25,and 4.26 are analogous to Facts 4.21,4.22,and 4.23 respectively and can be proved in a similar way.

Fact 4.24 *If CL_j is (entirely) below the row y_c of M, then the CL_j-restriction of c' has all its zeroes in its tail TL_j.*

Fact 4.25 *If tail of TL_j intersects the row y_c of M, then CL_j-restriction of c' has all its zeroes in $TL_j \cup RL_{j,1}$.*

Fact 4.26 *If $jk + i = y_c$, $1 \leq i \leq k$, then the CL_j-restriction of c' contains zeroes only in the $RL_{j,i} \cup RL_{j,i+1}$.*

Corollary 4.27 *Let c be any k-disturbed configuration of R over $\{0,1\}$. Let $j = \lceil y_c/k \rceil$ and $i = (y_c \bmod k) + 1$. The configuration $c' = P_1P_2(c)$ has the following properties:*

1. *All displaced ones of c' are in $COL_{n_2} \cup RR_{j-1,k} \cup RR_{j,i} \cup RR_{j,i-1}$. The number of ones in $RR_{j-1,k} \cup RR_{j,i-1}$ is at most $k-1$.*
2. *All displaced zeroes of c' are in $COL_1 \cup RL_{j+1,1} \cup RL_{j,i} \cup RL_{j,i+1}$. The number of zeroes in $RL_{j+1,1} \cup RL_{j,i+1}$ is at most $k-1$.*

Proof. The property 1 follows from Facts 4.21, 4.22 and 4.23. Analogously the property 2 follows from Facts 4.24, 4.25 and 4.26. □

4.3.3 Phase 3

The aim of the third phase is to move all the displaced ones above ROW_{y_c} into $ROW_{y_c-1} \cup COL_{n_2}$ and all the zeroes below ROW_{y_c} into $ROW_{y_c+1} \cup COL_1$.

For this purpose we consider the unions $F_{j,i}$ of the subsets $(CL_j \setminus RL_{j,1}) \cap COL_{i+1}$ with the singletons $RL_{j+1,1} \cap COL_{k-i+1}$, for the displaced zeros in the left clusters, and the unions $G_{j,i}$ of $(CR_j \setminus RR_{j,k}) \cap COL_{n_2-k+i}$ with the singletons $RR_{j-1,k} \cap COL_{n_2-i}$ (see the middle part of Fig. 11).

Below we define the functions $f_{j,i}$ (respectively $g_{j,i}$) such that $f_{j,i}(s)$ (respectively $g_{j,i}(s)$) denotes the sth register of $F_{j,i}$ (respectively, of $G_{j,i}$).

For $0 \leq j \leq \lceil n_1/k \rceil$, for $1 \leq i \leq k-1$ let the $f_{j,i}$ and $g_{j,i}$ be mapping functions over $\{1, \ldots, k\}$ defined as follows.

$$f_{j,i}(x) = \begin{cases} jkn_2 + xn_2 + 1 + i & \text{if } x < k \ , \\ jkn_2 + xn_2 + k - i + 1 & \text{if } x = k \ , \end{cases}$$

$$g_{j,i}(x) = \begin{cases} (j-1)kn_2 + (x-1)n_2 - i & \text{if } x = 1 \ , \\ (j-1)kn_2 + (x-1)n_2 - k + i & \text{if } x > 1 \ . \end{cases}$$

Note that $f_{j,i}(\{1, \ldots, k-1\}) \cap R = (CL_j \setminus RL_{j,1}) \cap COL_{i+1}$ and $f_{j,i}(\{k\}) \cap R = RL_{j+1,1} \cap COL_{k-i+1}$. Analogously, $g_{j,i}(\{2, \ldots, k\}) \cap R = (CR_j \setminus RR_{j,k}) \cap COL_{n_2-k+i}$ and $g_{j,i}(\{1\}) \cap R = RR_{j-1,k} \cap COL_{n_2-i}$.

For $0 \leq j \leq \lceil n/k \rceil$ and $1 \leq i \leq k-1$, we have

$$F_{j,i} = f_{j,i}(\{1, \ldots, k\}) \cap R$$

and

$$G_{j,i} = g_{j,i}(\{1, \ldots, k\}) \cap R.$$

The third phase P_3 is defined as the R-restriction of the union of the $f_{j,i}$-mappings of INS_k^0 and the $g_{j,i}$-mappings of INS_k^1, for all j, $0 \leq j \leq \lceil n/k \rceil$ and for all i, $1 \leq i \leq k-1$.

Claim 4.28 *Let c be a k-disturbed configuration of R over $\{0,1\}$. Let $j = \lceil y_c/k \rceil$. Let $c' = P_1P_2(c)$. Then*

- *The $G_{j,i}$-restriction of c' has at most one one above ROW_{y_c} and the sequence corresponding to $G_{j,i}$-restriction of c' is either sorted or differs from the sorted sequence only at the first position.*
- *The $F_{j,i}$-restriction of c' has at most one zero below ROW_{y_c} and the sequence corresponding to $F_{j,i}$-restriction of c' is either sorted or differs from the sorted sequence only at the last position.*

Proof. The claim follows from definitions of $F_{j,i}$ and $G_{j,i}$. To see the property for $G_{j,i}$, note that if the first register of the $G_{j,i}$ and any of its remaining registers both contain the ones, then the sum of the numbers of ones in the corresponding rows of the right clusters must be greater than $k-1$. By Fact 4.20, it is possible only if the second register is below ROW_{y_c-1}. The part of the sequence in the registers $g_{j,i}(2)$ through $g_{j,i}(k)$ is sorted, since the cluster CR_j is sorted. See Fig. 11. □

Lemma 4.29 *Let c be a k-disturbed configuration of R over $\{0,1\}$. Let $c' = P_1P_2P_3(c)$. Then c' has following properties:*

1. *All registers that are above ROW_{y_c-1} and on the left side of COL_{n_2} contain only zeroes.*
2. *All registers of ROW_{y_c-1} on the left side of COL_{n_2-k+1} contain only zeroes.*
3. *All registers below ROW_{y_c+1} and on the right side of COL_1 contain only ones.*
4. *All registers of ROW_{y_c+1} on the right side of COL_k contain only ones.*

Proof. We prove only the Properties 1 and 2. The Properties 3 and 4 are dual and can be proved in an analogous way.

By Fact 4.19, all the ones above ROW_{y_c} in $P_1P_2(c)$ are in the k rightmost columns. By Claim 4.28 the sequence corresponding to $G_{j,i}$-restriction of $P_1P_2(c)$ has at most one one above ROW_{y_c} and has the structure that can be sorted by INS_k^1. Property 1 follows from the fact that each $G_{j,i}$-restriction of $P_1P_2(c)$ is sorted by the corresponding $g_{j,i}$-mapping of INS_k^1.

By the Fact 4.19, the part of ROW_{y_c-1} on the left side of COL_{n_2-k+1} contains no ones in $P_1P_2(c)$. Property 2 follows from the definition of P_3: All the comparators of P_3 with the second registers in the ROW_{y_c-1} on the left side of COL_{n_2-k+1} have their first registers in the leftmost k columns above ROW_{y_c}. By Fact 4.19, these registers must contain zeroes in $P_1P_2(c)$. Thus no such comparator can insert a one into ROW_{y_c-1}. □

Fig. 11 illustrates what happens during Phase 3 in the two right clusters that intersect the row y_c. The leftmost part displays the placement of the ones

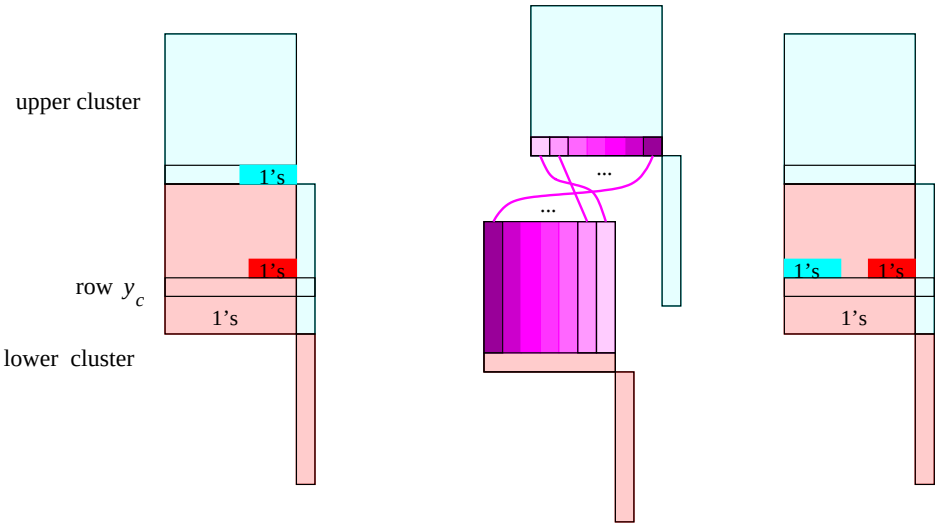


Figure 11: The configuration transformation during the phase 3

in the last row of the upper cluster and in the row of the lower cluster that is just above ROW_{y_c}. The rightmost part shows the placement of these ones after the application of P_3. The middle part shows how the registers from the last row of the upper cluster are grouped together in the phase P_3 with the columns of the next lower cluster.

4.3.4 Phase 4

Phase 4 is the core part of the construction. It demonstrates the technique of embedding the networks $I^1_{n_1}$ and $I^0_{n_1}$ in the matrix of registers M of size $n_1 \times n_2$ in such a way that after application of P_4 all the displaced elements are concentrated in at most three neighboring rows of M.

Trees of columns. Below we define the trees that will be used in description of the construction of P_4.

Definition 4.30 *Let T_d denote the tree with the edges labeled by positive integers, defined recursively as follows:*

1. *T_0 contains only a single isolated vertex (a root of T_0).*
2. *For $d > 0$, T_d is a tree constructed from two copies of T'_{d-1} (where T'_{d-1} is created from T_{d-1} by increasing the labels of all edges by one) by connecting the root of the first T'_{d-1} as the new child of the root of the second T'_{d-1} with a new edge labeled 1. The root of the second T'_{d-1} is a root of T_d.*

By a level *of a node in T_d we denote its distance from the root (i.e. the level of the root is 0, the level of any child of the root is 1, and so on). By $T_{d,t}$ we*

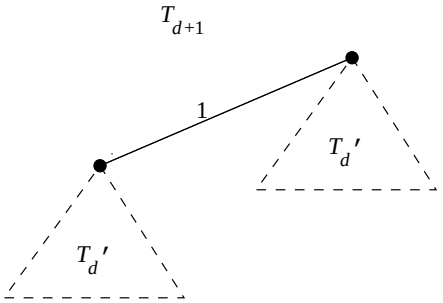


Figure 12: Construction of T_{d+1} from T'_d

denote the subtree of T_d consisting of the nodes with the levels less or equal to t. By $\alpha_{d,t}$ we denote the number of vertices of $T_{d,t}$.

Note that T_d is isomorphic with a binomial tree. Since the binomial tree has $\binom{d}{i}$ vertices on the ith level, we have $\alpha_{d,t} = \sum_{i=0}^{t} \binom{d}{i}$.

Let $T = T_{2\lceil \log n \rceil - 1, \lceil \log k \rceil}$. Let $\alpha = \alpha_{2\lceil \log n \rceil - 1, \lceil \log k \rceil}$. We can now define the value of n_2 (and thus of n_1): $n_2 = 2\alpha$.

Construction of P_4. We use the tree T to describe the construction of P_4. Let V be a set of vertices of T and let $v_0 \in V$ be a root of T.

We define two sets of columns:

$$CSET_1 = \{COL_i \mid 2 \leq i \leq n_2/2\}$$

and

$$CSET_0 = \{COL_i \mid n_2/2 < i \leq n_2 - 1\}.$$

By Lemma 4.29 and by the fact that $n_2 > 2k$ the following holds.

Claim 4.31 *Let c be a k-disturbed configuration over $\{0,1\}$. Let $c' = P_1P_2P_3(c)$. If $COL_i \in CSET_1$, then COL_i-restriction of c' has only zeroes above ROW_{y_c}. If $COL_i \in CSET_0$, then COL_i-restriction of c' has only ones below ROW_{y_c}.*

Let

$$\mathrm{col}_0 : V \to \{COL_1\} \cup CSET_0$$

be any bijection such that $\mathrm{col}_0(v_0) = COL_1$. Symmetrically, let

$$\mathrm{col}_1 : V \to \{COL_{n_2}\} \cup CSET_1$$

be any bijection such that $\mathrm{col}_1(v_0) = COL_{n_2}$. Let col_i^{-1} denote the reverse function of col_i, for $i \in \{0,1\}$.

The phase $P_4 = (P_{4,1}, \ldots, P_{4,4\lceil \log n \rceil - 2})$ is defined as follows:

- The columns $\{COL_{n_2}\} \cup CSET_1$ of M contain the embedding of $I^1_{n_1}$. The comparators of P_4 in this part of M are defined as follows:

- For each l, $1 \leq l \leq 2\lceil \log n \rceil - 1$, for each column $COL_i \in \{COL_{n_2}\} \cup CSET_1$ such that $\mathrm{col}_1^{-1}(COL_i) = v_0$ or the vertex $\mathrm{col}_1^{-1}(COL_i)$ is connected with its parent by an edge with a label less than l, there is a comparator (r_1, r_2) in $P_{4,2l-1}$ if and only if $\{r_1\} = ROW_{j_1} \cap COL_i$ and $\{r_2\} = ROW_{j_2} \cap COL_i$ and the lth layer of $I^1_{n_1}$ contains the comparator (j_1, j_2).
- For each l, $1 \leq l \leq 2\lceil \log n \rceil - 1$, for each column $COL_i \in \{COL_{n_2}\} \cup CSET_1$ such that the vertex $v_i = \mathrm{col}_1^{-1}(COL_i)$ is connected with its child v'_i by an edge with a label l, there is a comparator (r_1, r_2) in $P_{4,2l}$ if and only if $\{r_1\} = ROW_{j_1} \cap COL_i$ and $\{r_2\} = ROW_{j_2} \cap \mathrm{col}_1(v'_i)$, and the lth layer of $I^1_{n_1}$ contains the comparator (j_1, j_2).

- Analogously we define the embedding of $I^0_{n_1}$ in the columns $\{COL_1\} \cup CSET_0$.

The idea behind the construction is the following one: Consider the displaced ones in COL_{n_2}. We want to move them down to the row at least $y_c - 1$. Initially, COL_{n_2} contains at most k ones above ROW_{y_c} (displaced ones) and the columns from $CSET_1$ contain no ones above ROW_{y_c} (we say they are *clean*). Any displaced one falls down inside its column through the comparators in the odd layers of P_4 until it is blocked by another displaced one. In that case, the corresponding comparator in the next even layer of P_4 moves the blocked one to the register in some clean column to the same row that it would have reached if it had not been blocked. Note the clean column c' that can receive displaced ones from a column c in the layer $P_{4,2l}$ corresponds to a child of the vertex corresponding to c. On the other hand, c can move at most half of its displaced ones to c'. That means that the columns corresponding to the vertices on the level $\lceil \log k \rceil$ will receive at most one displaced one and will never have collisions between displaced elements. That is why we could clip $T_{2\lceil \log n \rceil - 1}$ to $T_{2\lceil \log n \rceil - 1, \lceil \log k \rceil}$ in our construction.

Lemma 4.32 *Let c be any k-disturbed configuration of R over $\{0,1\}$. Let $c' = P_1P_2P_3P_4(c)$. Then c' has only zeroes above the ROW_{y_c-1} and only ones below the ROW_{y_c+1}.*

Proof. We prove only that c' has no ones above ROW_{y_c-1}. The proof that c' has no zeroes below ROW_{y_c+1} is analogous.

Let $X_0 = COL_1 \cup \bigcup_{COL_i \in CSET_0} COL_i$. Let $X_1 = COL_{n_2} \cup \bigcup_{COL_i \in CSET_1} COL_i$. Note that each comparator of phase P_4 has both its registers either in X_0 or in X_1.

Since X_0-restriction of the configuration $c'' = P_1P_2P_3(c)$ has no ones above ROW_{y_c-1} and P_4 is a sequence of standard layers, there are also no ones above ROW_{y_c-1} in the X_0-restriction of c'.

Thus we have to show only that all the ones that are above the ROW_{y_c-1} in the X_1-restriction of c'' will be moved out of this region by P_4.

Claim 4.33 *Let COL_i be a column from $CSET_1$. Let t be the label of the edge connecting $\mathrm{col}_1^{-1}(COL_i)$ with its parent. Let $0 \leq t' < 2t$. Then there are no ones above ROW_{y_c} in the COL_i restriction of $P_{4,1} \ldots P_{4,t'}(c'')$*

Proof of the claim. By Claim 4.31, all columns of $CSET_1$ have only zeroes above ROW_{y_c} in configuration c''. The only column in the X_1-restriction of c'' that may have ones above ROW_{y_c} is COL_{n_2}. $P_{4,2t}$ is the first (and the only) layer of P_4 that has comparators with the second register in COL_i and the first register from outside COL_i (i.e. the only layer that can increase the number of ones in COL_i).

Claim 4.34 *Let COL_i be a column from $CSET_1 \cup \{COL_{n_2}\}$. Let l be the level of $\mathrm{col}_1^{-1}(COL_i)$ in T. Let m_t be the number of ones above ROW_{y_c} in the COL_i-restriction of the configuration $P_{4,1} \ldots P_{4,t}(c'')$. Then $\max\{m_t \mid 0 \leq t \leq 4\lceil \log n \rceil - 2\} \leq 2^{-l}k$.*

Proof of the claim. By induction on l. If $l = 0$, then $COL_i = COL_{n_2}$ and the number of ones in this column above ROW_{y_c} is never greater than $k = 2^{-0}k$. If $l > 0$, then the number of ones above ROW_{y_c} in the column COL_j such that $\mathrm{col}_1^{-1}(COL_j)$ is the parent of $\mathrm{col}_1^{-1}(COL_i)$, is never greater than $2^{-(l-1)}k$. There is only one layer $P_{4,t'}$ in the phase P_4 that contains comparators with the first register in COL_j and the second register in COL_i. All the remaining layers of P_4 contain comparators with either both registers in COL_i or with the second register outside COL_i. Before application of $P_{4,t'}$ there are no ones above ROW_{y_c} in COL_i. The comparator (r_1, r_2) of $P_{4,t'}$ can move a one from COL_j to COL_i if and only if the comparator (r_1, r_2') from the phase $P_{4,t'-1}$ with both registers in COL_j had ones in its both registers. There are at most $2^{-(l-1)}k/2 = 2^{-l}k$ such comparators.

Claim 4.35 *Let m be the number of ones above ROW_{y_c} in the COL_{n_2}-restriction of c''. (Note that $m \leq k$.) Let $s_1, \ldots, s_m$ be the increasing sequence of all their row positions. Let $c_{1,0}, \ldots, c_{m,0}$ be a sequence of configurations of $\{1, \ldots, n_1\}$ over $\{0, 1\}$, such that*

$$c_{i,0}(j) = \begin{cases} 1 & \text{if } j = s_i \text{ or } j \geq y_c \\ 0 & \text{otherwise.} \end{cases}$$

Let $c_{i,t}$ be a result of application of the subsequence of t initial layers of I_n^1 to the configuration $c_{i,0}$. Let $s_{i,t}$ be the (unique) number such that $s_{i,t} < y_c$ and $c_{i,t}(s_{i,t}) = 1$. Let $c_t'' = P_{4,1} \ldots P_{4,2t}(c'')$. Let m_t be a number of rows above ROW_{y_c} that contain ones in the X_1-restriction of c_t'', and let $s_{1,t}'', \ldots, s_{m_t,t}''$ be the increasing sequence of their indexes. Then

$$\{s_{1,t}, \ldots, s_{m,t}\} = \{s_{1,t}'', \ldots, s_{m_t,t}''\}.$$

Proof of the claim. By induction on t. The case for t_0 follows from the facts that each $s_{i,0} = s_i$ and all the ones above ROW_{y_c} in X_1-restriction

of c'' are in COL_{n_2}. Let $t > 0$. We have to show that if $\{s_{1,t}, \ldots, s_{m,t}\} = \{s''_{1,t}, \ldots, s''_{m_t,t}\}$, then $\{s_{1,t+1}, \ldots, s_{m,t+1}\} = \{s''_{1,t+1}, \ldots, s''_{m_{t+1},t+1}\}$. Note that either $s_{i,t+1} = s_{i,t}$ or there is an comparator $(s_{i,t}, s_{i,t+1})$ in the tth layer of $I^1_{n_1}$ and $s_{i,t+1} < y_c$. If there are any ones above ROW_{y_c} in the COL_i-restriction of c''_t, then $\mathrm{col}_1^{-1}(COL_i)$ either is a root of T or must be connected with its parent by an edge with the label not greater than t. By the definition of P_4, either:

- there are comparators (r_1, r_2) in $P_{4,2t+1}$ and (r_1, r'_2) in $P_{4,2t+2}$ such that
  - $\{r_1\} = COL_i \cap ROW_{s_{i,t}}$ and
  - $\{r_2\} = COL_i \cap ROW_{s_{i,t+1}}$ and
  - $\{r'_2\} = COL_j \cap ROW_{s_{i,t+1}}$,

  where $\mathrm{col}_1^{-1}(COL_j)$ is a child of $\mathrm{col}_1^{-1}(COL_i)$ connected with it by an edge with the label $t+1$, or
- there is only the comparator (r_1, r_2) in $P_{4,2t+1}$ and the level of $\mathrm{col}_1^{-1}(COL_i)$ in T is $\lceil \log k \rceil$.

By the Claim 4.34, in the second case there is at most single one above ROW_{y_c} in COL_i and it will be shifted by the comparator (r_1, r_2) from the $ROW_{s_{i,t}}$ to the $ROW_{s_{i,t+1}}$.

By the Claim 4.33, in the first case the one from r_1 is shifted to r_2 or to r'_2, since $P_{4,2t+1}(c''_t)(r'_2) = 0$.

Lemma 4.32 follows from the fact that by Lemma 4.12 $\{s_{1,t}, \ldots, s_{m,t}\} \subseteq \{y_c - 1\}$, for $t = 2\lceil \log n \rceil - 1$. □

4.3.5 Phase 5

By Lemma 4.32, after application of the phases $P_1P_2P_3P_4$ to a k-disturbed zero-one configuration c there are only zeroes above ROW_{y_c-1} and only ones below ROW_{y_c+1}. Let $c' = P_1P_2P_3P_4(c)$. $(ROW_{y_c-1} \cup ROW_{y_c})$-restriction of c' and $(ROW_{y_c} \cup ROW_{y_c+1})$-restriction of c' are k-disturbed, since c' is k-disturbed. The last phase P_5 is defined as follows:

$$P_5 = P'_5 P''_5 P'''_5,$$

where P'_5, P''_5 and P'''_5 are defined below.

Let L be a $\{1, \ldots, 2n_2\}$-restriction of the subsequence of layers of $S_{\lceil 2n_2/k \rceil, k}$. (Recall that this is the extended k-merge version of the Schimmler Starke 1-correction network.) Let P'_5 be the R-restriction of a union of the f_i-mappings of L, where $f_i(x) = 2n_2 i + x$.

Let M be a sequence of layers of BM_{2n_2} (the Batcher merging networks for two sorted sequences of length $2n_2$ stored in the registers $\{1, \ldots, 2n_2\}$ and $\{2n_2 + 1, \ldots, 4n_2\}$). Let P''_5 (respectively P'''_5) be the R-restriction of a union of the g_i-mappings (respectively g'_i-mappings) of M, where $g_i(x) = 4n_2 i + x$ (respectively $g'_i(x) = g_i(x) + 2n_2$).

The following lemma states that $N_{n,k}$ is a k-correction network.

Lemma 4.36 *Let c be any k-disturbed configuration of R over $\{0,1\}$. Then $c' = P_1P_2P_3P_4P_5(c)$ is sorted.*

Proof. The configuration $c'' = P_1P_2P_3P_4P'_5(c)$ has following properties:

- for $i \geq 1$, the $(ROW_{2i-1} \cup ROW_{2i})$-restriction of c'' is sorted (by Lemma 4.14).
- Let $i_0 = \lceil (y_c + 1)/2 \rceil$. Then $2(i_0 - 2) + 1 \leq y_c - 1$, so c'' has only zeroes above $ROW_{2(i_0-2)+1}$. (Since, by Lemma 4.32, all displaced elements of c'' are contained in $ROW_{y_c-1} \cup ROW_{y_c} \cup ROW_{y_c+1}$.) Similarly, $2i_0 \geq y_c + 1$, so c'' has only ones below ROW_{2i_0}.

It is enough to sort the fragment of c'' contained in the rows $2(i_0-2)+1$ through $2i_0$. Both $(ROW_{2i_0-3} \cup ROW_{2i_0-2})$-restriction of c'' and $(ROW_{2i_0-1} \cup ROW_{2i_0})$-restriction of c'' are sorted. Thus all we have to do is merge the subsequences contained in $ROW_{2i_0-3} \cup ROW_{2i_0-2}$ and $ROW_{2i_0-1} \cup ROW_{2i_0}$. We do not know the parity of i_0. If i_0 is even, then already $P''_5(c'')$ is sorted. Otherwise $P''_5(c'') = c''$ and $P'''_5(P''_5(c''))$ is sorted. □

4.3.6 Estimation of the depth of $N_{n,k}$

For any positive integer i, let m_i denote the depth of the Batcher merging network BM_i that merges two sequences of length i each, and let d_i denote the depth of the Batcher sorting network BS_i for input of size i. Then $m_i = 1 + \lceil \log i \rceil$ and $d_i = \frac{m_i \lceil \log i \rceil}{2}$.

The depth p_1 of the phase P_1 is equal to the depth of $S_{\lceil n_2/k \rceil, k}$. Thus

$$p_1 = d_k + m_k \left(2\lceil \log \lceil n_2/k \rceil \rceil - 1\right) = m_k \left(\frac{\lceil \log k \rceil}{2} + 2\lceil \log \lceil n_2/k \rceil \rceil - 1 \right)$$

$$\leq m_k \left((\log k)/2 + 2\log(n_2/k) + \frac{3}{2} \right) = m_k \left(2\log n_2 - \frac{3}{2}\log k + \frac{3}{2} \right)$$

$$\leq 2\log n_2 (1 + \lceil \log k \rceil) - \frac{3}{2}\log k - \frac{3}{2}\log^2 k + \frac{3}{2} + \frac{3}{2}\lceil \log k \rceil$$

$$\leq 2\log n_2 (1 + \lceil \log k \rceil) - \frac{3}{2}\log^2 k + 3.$$

Analogously the depth p'_5 of P'_5 (the depth of $S_{\lceil 2n_2/k \rceil, k}$) is not greater than

$$2\log(2n_2)(1 + \lceil \log k \rceil) - \frac{3}{2}\log^2 k + 3.$$

The depth p''_5 of P''_5 (and of P'''_5) is equal to the depth of BM_{2n_2}, so $p''_5 = 1 + \lceil \log(2n_2) \rceil$. Thus the depth of phase P_5 is equal to

$$p_5 = p'_5 + 2p''_5 \leq 2\log(2n_2)(1 + \lceil \log k \rceil) - \frac{3}{2}\log^2 k + 3 + 2 + 2\lceil \log(2n_2) \rceil$$

$$\leq 2\log(2n_2)(2+\lceil\log k\rceil) - \frac{3}{2}\log^2 k + 7.$$

The depth p_2 of P_2 is equal to the depth of BS_{k^2}:

$$p_2 = d_{k^2} = m_{k^2}\frac{\lceil\log(k^2)\rceil}{2} \leq 2(1+\log k)^2 = 2\log^2 k + 4\log k + 2.$$

The depth p_3 of the phase P_3 is equal to the depth of INS_k^1 (or INS_k^0):

$$p_3 = \lceil\log k\rceil \leq \log k + 1.$$

The depth p_4 of P_4 is twice the depth of $I_{n_1}^1$ (or $I_{n_1}^0$).

$$p_4 = 4\lceil\log(n/n_2)\rceil - 2 \leq 4\log n - 4\log n_2 + 2.$$

The depth of P can be estimated as follows:

$$p = p_1 + p_2 + p_3 + p_4 + p_5 \leq \left(2\log n_2(1+\lceil\log k\rceil) - \frac{3}{2}\log^2 k + 3\right)$$
$$+(2\log^2 k + 4\log k + 2) + (\log k + 1) + (4\log n - 4\log n_2 + 2)$$
$$+\left(2\log(2n_2)(2+\lceil\log k\rceil) - \frac{3}{2}\log^2 k + 7\right)$$

Finally

$$p \leq 4\log n + 4\log n_2\left(\frac{1}{2} + \lceil\log k\rceil\right) - \log^2 k + 7\log k + 21 \tag{2}$$

We have to estimate n_2. Recall that n_2 (the number of columns of the matrix of registers M) is twice the number of vertices of the tree T. On the other hand T is a subtree of the binomial tree $T_{2\lceil\log n\rceil-1}$, consisting of the vertices on the levels not greater than $\lceil\log k\rceil$. The number of the vertices on the ith level of T_m is $\binom{m}{i}$. Thus

$$n_2 = 2\cdot\sum_{i=0}^{\lceil\log k\rceil}\binom{2\lceil\log n\rceil - 1}{i} \tag{3}$$

Lemma 4.37 *(due to Marek Piotrów) If* $n \geq 256$ *and* $3 \leq k \leq \frac{1}{2}n^{\frac{1}{3+\log\log n}}$, *then* $\log n_2 \leq \lceil\log k\rceil(\log\log n + 2)$ *and* $n_2 \leq n/k$.

Proof. By an easy induction one can prove that

$$\sum_{i=0}^{j-1}\binom{m}{i} \leq \binom{m}{j} \tag{4}$$

for $m \geq 2$ and $j \leq \frac{m+1}{3}$. In our case

$$\lceil\log k\rceil \leq \left\lceil\frac{\log n}{3+\log\log n}\right\rceil \leq \tfrac{2}{3}\lceil\log n\rceil \leq \frac{(2\lceil\log n\rceil - 1) + 1}{3}.$$

Therefore, we can apply the inequality (4) to the sum (3) without the last term and obtain

$$n_2 \le 4\binom{2\lceil \log n\rceil - 1}{\lceil \log k\rceil}. \tag{5}$$

By Stirling formula, we have the following well-known upper bound:

$$\binom{m}{j} \le \frac{1}{\sqrt{2\pi j}}\left(\frac{me}{j}\right)^j.$$

Applying this to (5) and taking logarithm of both sides we get

$$\log n_2 \le \left(\tfrac{3}{2} - \tfrac{1}{2}\log(\pi\lceil \log k\rceil)\right) + \lceil \log k\rceil \left(\log\log n + \log\frac{e(2+1/\log n)}{\lceil \log k\rceil}\right)$$

Due to our assumption about k and n, $\lceil \log k\rceil \ge 2$ and $\log n \ge 8$, and therefore the expression in the first parenthesis is bounded by 0.2 and

$$\log\frac{e(2+1/\log n)}{\lceil \log k\rceil} \le 1.6.$$

The first part of lemma follows. The second one is a simple consequence of the first part and the upper bound on k:

$$\lceil \log k\rceil \le 1 + \log k \le \frac{\log n}{3 + \log\log n}$$

and

$$\log n_2 \le \lceil \log k\rceil(\log\log n + 3) - \log k \le \log\frac{n}{k}.$$

□

Lemma 4.37 shows that the construction is correct: the required number of columns does not exceed the total number of registers and there are at least k rows.

If $n \ge 256$ and $3 \le k \le \frac{1}{2}n^{\frac{1}{3+\log\log n}}$, then by Lemma 4.37 and by the estimation (2) we have:

$$p \le 4\log n + 4\lceil \log k\rceil(\log\log n + 2)\left(\frac{1}{2} + \lceil \log k\rceil\right) - \log^2 k + 7\log k + 21$$

$$\le 4\log n + 4\lceil \log k\rceil^2 \log\log n + 2\lceil \log k\rceil \log\log n + 7\log^2 k + 11\log k + 33$$

Thus

$$p = 4\log n + O(\log^2 k \log\log n).$$

5 Periodic correction networks

In this section we consider the problem of sorting k-disturbed sequences with the periodic networks of a constant depth. We start in Section 5.1 with a presentation of a simple periodic 1-correction network of depth 4 that works in $O(\log n)$ iterations, and then in Section 5.2 we present a periodic k-correction network of depth 8 that works in $O(k + \log n)$ iterations.

5.1 A simple periodic 1-correction network

In this section we define a simple 1-correction network H_l on $2^l \cdot 2l$ registers $\{0, \ldots, 2^l \cdot 2l - 1\}$.

We start with a definition of two auxiliary networks G_l and G'_l. Let $N_l = 2^l(l+1)$.

Definition 5.1 *Let l be a positive integer. Let*

$$g_l : \{0, \ldots, l\} \times \{0, \ldots, 2^l - 1\} \to \{0, \ldots, N_l - 1\}$$

be a bijection defined as follows:

$$g_l(x, y) = x + (l+1)y.$$

We assume that the registers are arranged in a matrix, where the register $g_l(x, y)$ is placed in column x and row y. (The rows and columns are numbered from zero.)

Definition 5.2 *For a positive integer l, we define a network $G_l = CN(N_l, 4, R, L)$, where $R = \{0, \ldots, N_l - 1\}$, $L = (L_0, L_1, L_2, L_3)$, and (see Fig. 13):*

- $L_0 = \{(g_l(x, y), g_l(x+1, y + 2^{l-x-1})) \mid x \text{ is even}, \ 0 \le x < l, \ 0 \le y < 2^l - 2^{l-x-1}\}$,
- $L_1 = \{(g_l(x, y), g_l(x+1, y)) \mid x \text{ is even}, \ 0 \le x < l, \ 0 \le y < 2^l\}$,
- $L_2 = \{(g_l(x, y), g_l(x+1, y + 2^{l-x-1})) \mid x \text{ is odd}, \ 0 \le x < l, \ 0 \le y < 2^l - 2^{l-x-1}\}$,
- $L_3 = \{(g_l(x, y), g_l(x+1, y)) \mid x \text{ is odd}, \ 0 \le x < l, \ 0 \le y < 2^l\}$.

We define a network G'_l that is symmetrical to G_l:

Definition 5.3 *For a positive integer l and $R = \{0, \ldots, N_l - 1\}$ we define a network*

$$G'_l = CN(N_l, 4, R, (L'_0, L'_1, L'_2, L'_3)),$$

such that for each i, $0 \le i \le 3$,

$$L'_i = \{(r_1, r_2) \mid (N_l - 1 - r_2, N_l - 1 - r_1) \in L_i\},$$

where L_i is the ith layer of the network G_l.

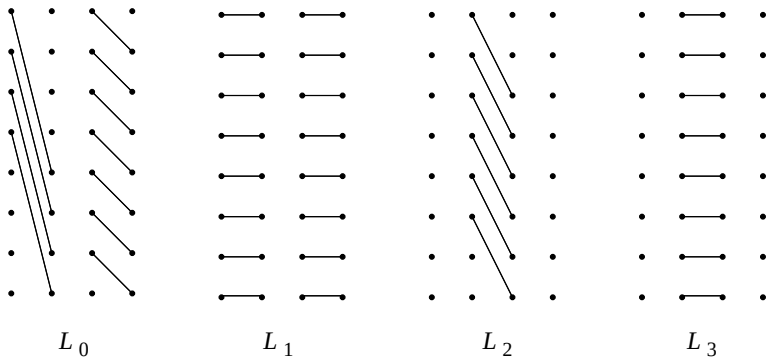


Figure 13: The layers of G_3.

We arrange the registers of H_l into a matrix that contains in column x and row y the register $h_l(x, y)$, where the function h_l is defined as follows.

Definition 5.4 *Let l be a positive integer. Let*

$$h_l : \{0, \ldots, 2l-1\} \times \{0, \ldots, 2^l - 1\} \to \{0, \ldots, 2^l \cdot 2l - 1\}$$

be a bijection defined as follows:

$$h_l(x, y) = x + 2ly.$$

Below we define two functions m_l and m'_l that are used for mapping the layers of respectively G_l and G'_l into H_l.

Definition 5.5 *Let*

$$m_l : \{0, \ldots, N_l - 1\} \to h_l(\{l-1, \ldots, 2l-1\} \times \{0, \ldots, 2^l - 1\})$$

and

$$m'_l : \{0, \ldots, N_l - 1\} \to h_l(\{0, \ldots, l\} \times \{0, \ldots, 2^l - 1\})$$

(where $h_l(X \times Y) = \{h_l(x, y) \mid x \in X,\ y \in Y\}$) be two mapping functions defined as follows. For each $0 \leq x \leq l$, for each $0 \leq y < 2^l$,

$$m_l(g_l(x, y)) = h_l(x + l - 1, y)$$

and

$$m'_l(g_l(x, y)) = h_l(x, y).$$

The network H_l is defined as follows.

Definition 5.6 *For a positive integer l we define the network $H_l = CN(2^l \cdot 2l, 4, R, L'')$ (see Fig. 14), where $R = \{0, \ldots, 2^l \cdot 2l - 1\}$, $L'' = (L''_0, L''_1, L''_2, L''_3)$, and:*

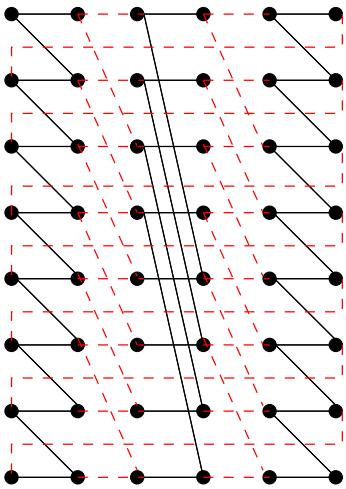


Figure 14: The network H_3. The layers L''_0 and L''_1 are drawn with the solid lines and the layers L''_2 and L''_3 are drawn with the dashed lines.

- *For each* $t \in \{0,1,2,3\}$*, the layer* L''_t *contains the union of the* m_l*-mapping of* L_t *and the* m'_l*-mapping of* L'_t*, where* L_t *and* L'_t *are the* t*th layers of* G_l *and* G'_l *respectively, and*
- *the layer* $L''_{1+2(l \bmod 2)}$ *contains additionally the set of comparators* $\{(h_l(2l-1, y), h_l(0, y+1)) \mid 0 \leq y < 2^l - 1\}$*, and*
- *there are no other comparators in* H_l*.*

Note that the corresponding mappings of L_t and L'_t, for $t \in \{0,1\}$ are not disjoint. (The two middle columns of H_l are in the images of both mappings.) However, the definition is correct because the comparators from the two mappings either are identical or contain no common registers.

Note that the layers of H_l are symmetrical (i.e. there is a comparator (i,j) in the layer L''_t if and only if there is a comparator $(n-1-j, n-1-i)$ in the same layer, where $n = 2^l \cdot 2l$ is the number of registers).

Lemma 5.7 *There exist a constant* d *such that for any* $l > 0$*, the network* H_l *sorts any* 1*-disturbed configuration in* dl *iterations.*

Proof. Let c be a 1-disturbed zero-one configuration on the registers of H_l. Let z denote the number of zeroes in c. Let $y'_c = \lfloor \frac{z}{2l} \rfloor$. That is, y'_c is the index of the first row of H_l that intersects the ones area. (The rows are numbered from zero.) Let r_t be the first register containing a one after t steps of computation of H_l (i.e. after application of the sequence of layers $(L''_0 \ldots L''_3)^{\lfloor t/4 \rfloor} L''_0 \ldots L''_{t \bmod 4}$). Let t' be the minimal t such that either $r_t \geq h_l(0, y'_c)$ (i.e. the displaced 1 is already in the row y'_c) or $r_t = h_l(l-1, y)$ for some y (i.e. r_t is in the column $l-1$).

Claim 5.8 $t' \leq 4l$.

The claim follows from the fact that there is at most one one above the row y'_c, and as long as it is above the row y'_c it is shifted every two steps to the next column on the right side or to the column 0 if it is in the column $2l-1$. (We say that the 1 is moved to the next (*modulo* $2l$) column on the right side.)

Let t'' be a minimal step number t such that $r_t \geq h_l(0, y'_c) - 1$.

Claim 5.9 $t'' \leq t' + 2l$.

If $r_{t'} \geq h_l(0, y'_c)$, then $t'' \leq t'$ and the claim holds. Otherwise, assume that $r_{t'}$ is in the column $l-1$ above the row y'_c. The first layer that can move a single displaced one from the register $r_{t'} = h_l(l-1, y)$ is L''_0 or L''_1. If the distance between the row y and the row y'_c is greater than 2^{l-1}, then the layer L''_0 moves the displaced one to the row $y + 2^{l-1}$ in the column l, otherwise L''_1 moves the displaced one to the register $h_l(l, y)$. In either case the distance between the displaced element and the row y'_c is not greater than 2^{l-1}. In a similar way, it can be shown by induction that after the step $t' + 2t$, where $1 \leq t \leq l$, the displaced one is in the column $l - 1 + t$ and the distance between its row and the row y'_c is at most 2^{l-t}.

Claim 5.10 *After* $6l$ *steps the configuration is at most* $2l+2$*-dirty.*

The claim follows from the fact that the network is symmetrical, and after $6l$ steps the index of the first register that contains a one is at least $h_l(0, y'_c) - 1$ and the index of the last register that contains a zero is at most $h_l(2l-1, y'_c) + 1$.

Now the lemma follows directly from Claim 5.10 and Lemma 3.4. □

For an arbitrary $n > 0$, we can construct a 1-correction periodic network of depth 4 for input sequences of size n as the $\{0, \ldots, n-1\}$ restriction of the network $H_{l'}$, where l' is the minimal l such that $2^l \cdot 2l \geq n$. Note that $l' \in O(\log n)$.

It is not clear how does the network H_l work for the k-disturbed configuration for the larger values of k. It can be shown easily, by considering each displaced element separately that the upper bound on the time needed for sorting such a configuration is $O(kl)$.

In the next section we construct a periodic network of a constant depth that sorts any k-disturbed configuration in $O(\log n + k)$ iterations.

5.2 Periodic k-correction network

The main problem with the k-disturbed sequences in the network H_l, for greater values k, is that a displaced one that starts falling in the column $l-1$ and is blocked by the other displaced one, may need a full rotation through all the columns to get another chance of falling down to the proper row. In this section we overcome to some extend this problem.

The layers $B_{l,i}$ and $B'_{l,i}$, defined below, will be used in the description of the k-correction network.

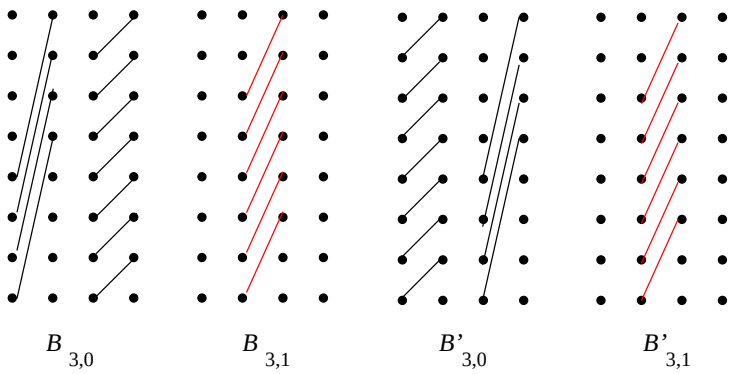


Figure 15: The layers $B_{3,i}$ and $B'_{3,i}$.

Definition 5.11 *For $l > 0$, we define four layers $B_{l,0}$, $B_{l,1}$, $B'_{l,0}$, $B'_{l,1}$ on the registers $\{0, \ldots, N_l - 1\}$ as follows (see Fig. 15):*

- $B_{l,0} = \{(g_l(x+1, y), g_l(x, y + 2^{l-x-1})) \mid x \text{ is even}, \ 0 \leq x < l, \ 0 \leq y < 2^l - 2^{l-x-1}\}$,
- $B_{l,1} = \{(g_l(x+1, y), g_l(x, y + 2^{l-x-1})) \mid x \text{ is odd}, \ 0 \leq x < l, \ 0 \leq y < 2^l - 2^{l-x-1}\}$,
- $B'_{l,0} = \{(r_1, r_2) \mid (N_l - 1 - r_2, N_l - 1 - r_1) \in B_{l,0}\}$,
- $B'_{l,1} = \{(r_1, r_2) \mid (N_l - 1 - r_2, N_l - 1 - r_1) \in B_{l,1}\}$.

Definition 5.12 *For $l > 0$ and the register sequence $R = \{0, \ldots, N_l - 1\}$ we define F_l and F'_l as follows (see Fig. 16):*

- $F_l = CN(N_l, 4, R, (L_1, B_{l,0}, L_3, B_{l,1}))$, *where L_1 and L_3 are the layers introduced in Definition 5.2.*
- $F'_l = CN(N_l, 4, R, (L'_1, B'_{l,0}, L'_3, B'_{l,1}))$, *where L'_1 and L'_3 are layers introduced in Definition 5.3.*

Now we can describe our main network. Let $l > 1$ and $w' > 1$ be integers. Let $w = 2(l + 1 + w')$, $h = 2^l$ and $n = wh$. Let $R = \{0, \ldots, n-1\}$. We define a function

$$r : \{0, \ldots, w-1\} \times \{0, \ldots, h-1\} \to \{0, \ldots, n-1\}$$

by

$$r(x, y) = x + wy.$$

We arrange the registers in a matrix where the register $r(x, y)$ is placed in column x and row y.

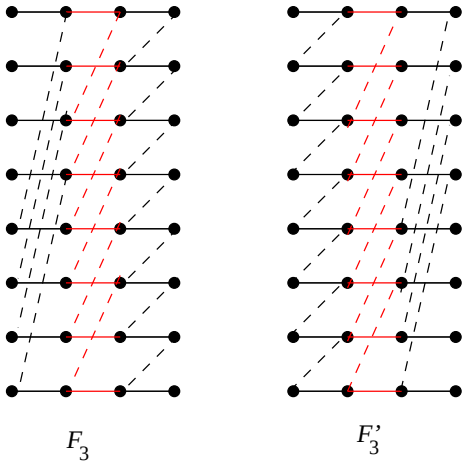


Figure 16: The networks F_3 and F_3': the layers $B_{3,0}$, $B_{3,1}$ of F_3 and the layers $B'_{3,0}$, $B'_{3,1}$ of F_3' are drawn with dashed lines.

We define the following two mapping functions m and m' that are used for placing the layers of F_l and F_l' on the registers of our network. For $0 \leq x \leq l$, $0 \leq y \leq 2^l - 1$,

$$m(g_l(x,y)) = r(w/2 + x, y)$$

and

$$m'(g_l(x,y)) = r(w/2 - l - 1 + x, y),$$

where g_l is the function defined in Definition 5.1.

Below we define our main network $P_{l,w'}$. For this purpose, we describe first auxiliary sequences of layers $Y = (Y_0, Y_1, Y_2, Y_3)$ and $J = (J_0, J_1, J_2, J_3)$.

Let (A_0, A_1, A_2, A_3) denote the sequence of layers of F_l and (A'_0, A'_1, A'_2, A'_3) denote the sequence of layers of F_l'. Let $Y = (Y_0, Y_1, Y_2, Y_3)$ be a sequence of layers, such that Y_t is the union of the m-mapping of A_t and the m'-mapping of A'_t (see Fig. 17). Note that in the matrix presentation (i.e. when the register $r(x,y)$ is placed in column x and row y) Y contains the mappings of the layers of F_l' and F_l, where the mapping of F_l' is placed at the columns $w/2 - l - 1, \ldots, w/2 - 1$ (where the columns are numbered from zero to $w-1$) and the mapping of F_l is placed at the columns $w/2, \ldots, w/2 + l$. Note that the layer Y_0 contains only comparators of the form $(r(x,y), r(x+1,y))$ while the layer Y_1 contains comparators of the form $(r(x+1, y_1), r(x, y_2))$, where the parity of x is the same as the parity of $w/2$. The layer Y_2 contains only comparators of the form $(r(x,y), r(x+1,y))$, while the layer Y_3 contains comparators of the form $(r(x+1, y_1), r(x, y_2))$, where the parity of x is the same as the parity of $w/2 + 1$.

Let $J = (J_0, J_1, J_2, J_3)$ be a sequence of layers over R defined as follows (see Fig. 18):

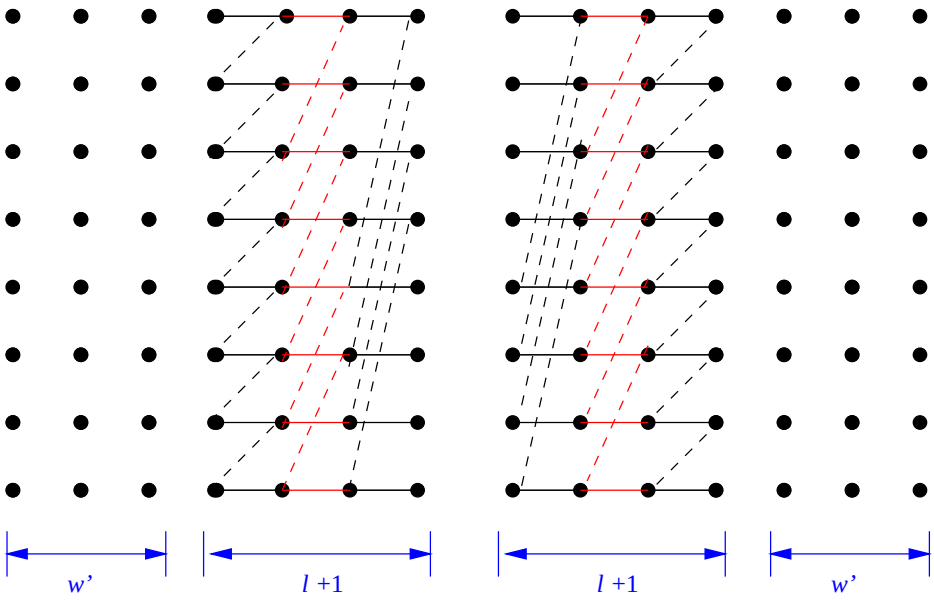


Figure 17: The layers Y_i for $l = 3$ and $w' = 3$. The layers Y_0 and Y_2 are drawn with solid lines and the layers Y_1 and Y_3 are drawn with the dashed lines.

- $J_0 = \{(r(x,y), r(x+1,y)) \mid 0 \le y < h,\ (x + \frac{w}{2}) \bmod 2 = 0,\ (0 \le x < w/2 - l - 1 \text{ or } w/2 + l \le x < w - 1)\}$,
- $J_1 = \{(r(x+1,y), r(x,y+1)) \mid 0 \le y < h - 1,\ (x + \frac{w}{2}) \bmod 2 = 0,\ (0 \le x < w/2 - l - 1 \text{ or } w/2 + l \le x < w - 1)\}$,
- $J_2 = \{(r(x,y), r(x+1,y)) \mid 0 \le y < h,\ (x + \frac{w}{2}) \bmod 2 = 1,\ (0 \le x < w/2 - l - 1 \text{ or } w/2 + l \le x < w - 1)\}$,
- $J_3 = \{(r(x+1,y), r(x,y+1)) \mid 0 \le y < h - 1,\ (x + \frac{w}{2}) \bmod 2 = 1,\ (0 \le x < w/2 - l - 1 \text{ or } w/2 + l \le x < w - 1)\}$.

Note that the only columns that contain the registers used by the comparators from both J and Y are the columns $w/2 - l - 1$ and $w/2 + l$.

The layers M_0, M_1, M_2, and M_3 that are defined below are presented on Fig. 21 for the case $l = 3$ and $w' = 3$. We define the sequence of layers of $P_{l,w'}$ as $M = (M_0, M', M_1, M', M_2, M', M_3, M')$, where

- $M' = \{(r(x,y), r(w-1-x,y)) \mid 0 \le x < w/2\}$ (see Fig. 19),
- for each t, $0 \le t \le 3$, the layer M_t contains $Y_t \cup J_t$, and
- for (the unique) $t \in \{0, 2\}$, such that J_t does not contain comparators with registers from the leftmost and the rightmost column, M_t contains the comparators $\{(r(w-1,y), r(0,y+1)) \mid 0 \le y < h - 1\}$, and

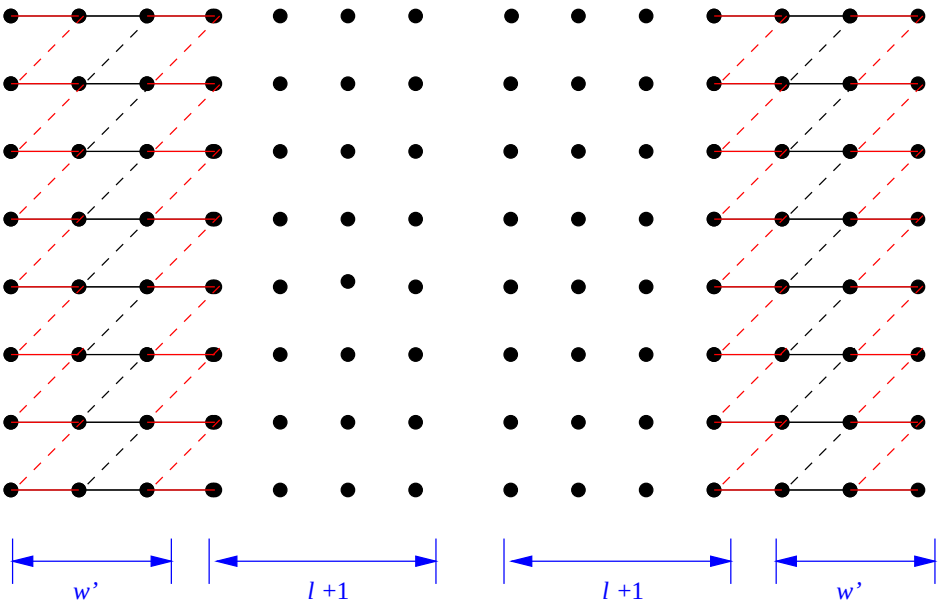


Figure 18: The layers J_i for $l = 3$ and $w' = 3$. The layers J_0 and J_2 are drawn with solid lines and the layers J_1 and J_3 are drawn with the dashed lines.

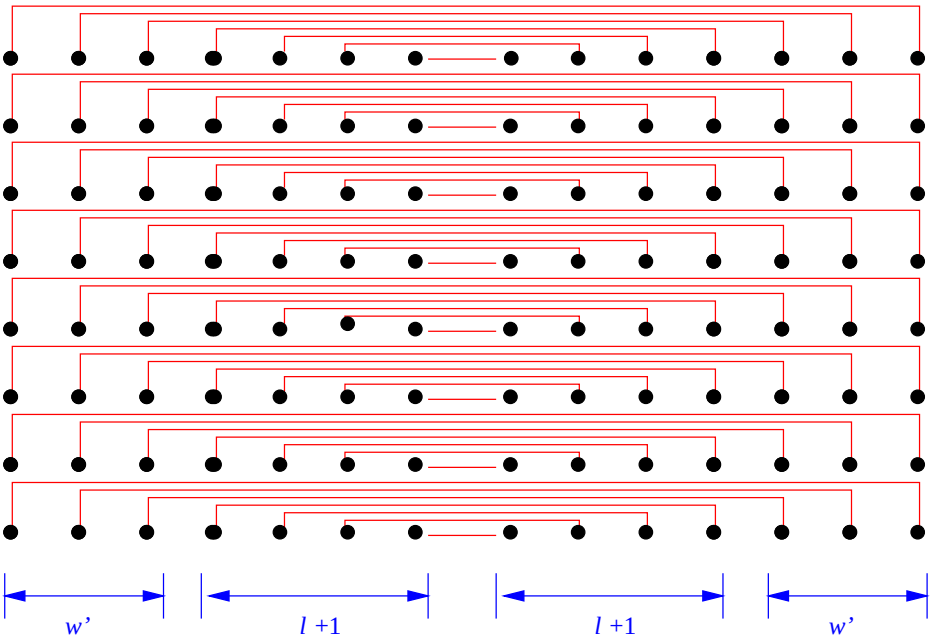


Figure 19: The layer M' (the *left-right* comparators) for $l = 3$ and $w' = 3$.

- there are no other comparators in the layers of M.

We call the layers M_0 and M_2 and their comparators *horizontal*. (Despite that the comparators of the form $(r(w-1,y), r(0,y+1))$ are slightly slanted, they are also called horizontal.)

M_1 and M_3 are called *back-jump* layers, and the comparators from these layers are called *back-jump* comparators.

We call the layers M' *left-right* layers and their comparators *left-right* comparators.

Observe that $P_{l,w'}$ has following property:

Fact 5.13 *The horizontal comparators together with the left-right comparators between columns $w/2-1$ and $w/2$, are all comparators of the odd-even transposition network on R.*

Note that the layer M_0 contains only comparators of the form $(r(x,y), r((x+1) \bmod w, y'))$ while the layer M_1 contains comparators of the form $(r((x+1) \bmod w, y_1), r(x,y_2))$, where the parity of all x's is the same as the parity of $w/2$. The layer M_2 contains only comparators of the form $(r(x,y), r((x+1) \bmod w, y'))$ while the layer M_3 contains comparators of the form $(r((x+1) \bmod w, y_1), r(x,y_2))$, where the parity of all x's is the same as the parity of $w/2+1$.

The only pairs of consecutive (modulo w) columns that are not connected by the back-jump comparators is $w/2-1$, $w/2$, and $w-1$, 0.

Fig. 20 presents a schematic view of the network $P_{l,w'}$. The subsets of registers used by the comparators from Y (respectively from J) are drawn as the boxes labeled by the letter Y (respectively J). The comparators that are neither contained in the layers of Y nor in the layers of J are drawn as the arrows. The only left-right comparators depicted are the comparators between the two middle columns.

Fig. 21 presents the network $P_{l,w'}$ for $l=3$ and $w'=3$ (without the left-right comparators).

Fig. 22 presents $P_{3,3}$ in a *folded* state (i.e. the left half of the network has been rotated 180 degrees around the central vertical axis in such a way that the mirror reflection of it is behind the right half of the network). The figure also presents the comparators between the columns $w-1$ and 0 (with the left-right comparators drawn as dotted arrows).

We partition the set of registers R into the *left set* $S_0 = \{r(x,y) \mid 0 \leq x \leq w/2-1\}$ and the *right set* $S_1 = \{r(x,y) \mid w/2 \leq x \leq w-1\}$. The members of S_0 (respectively S_1) are called *left* (respectively *right*) registers. For any register $r = r(x,y)$, we define a *shadow* of r, denoted by $\mathrm{shd}(r)$, as the register $r(w-1-x,y)$. Note that in the folded state, the shadow of each register r is placed in the same place as r, and that M' contains comparators of the form $(\mathrm{shd}(r), r)$. For any subset $X \subseteq S$ we define the shadow of X as $\mathrm{shd}(X) = \{\mathrm{shd}(r) \mid r \in X\}$.

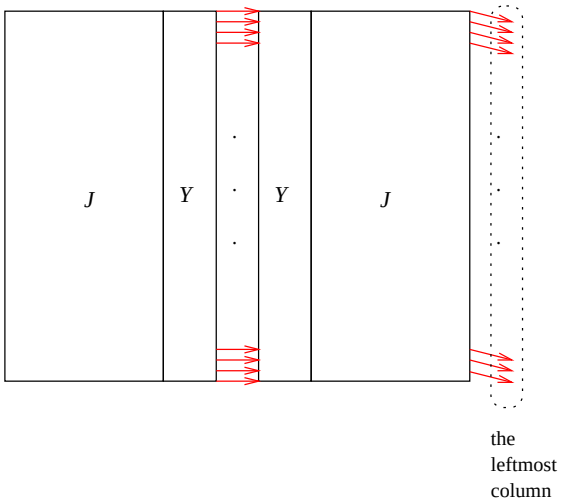


Figure 20: The layers of $P_{l,w'}$.

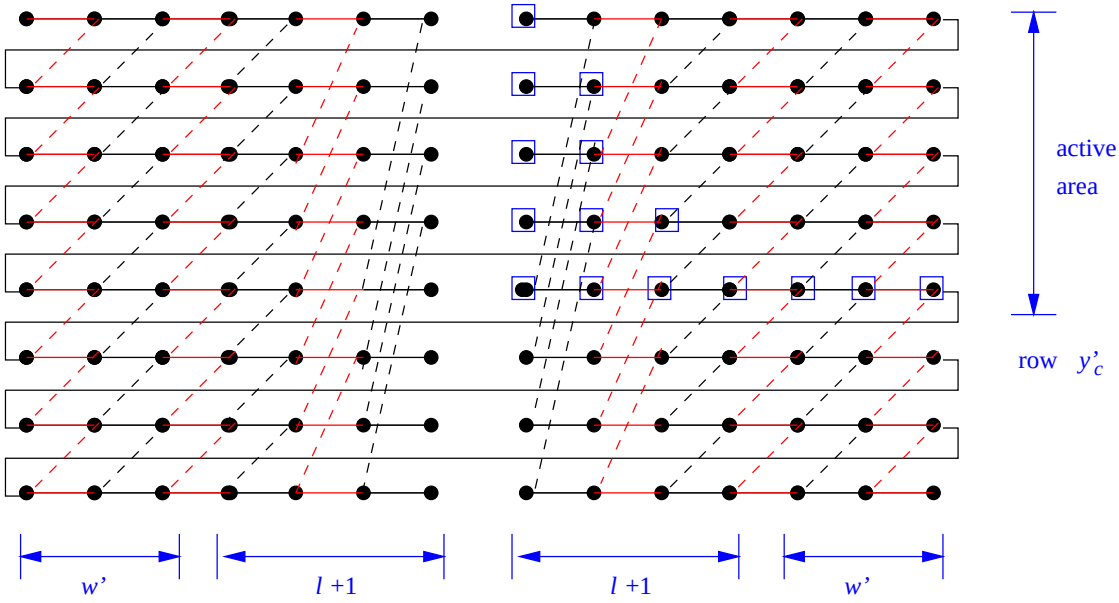
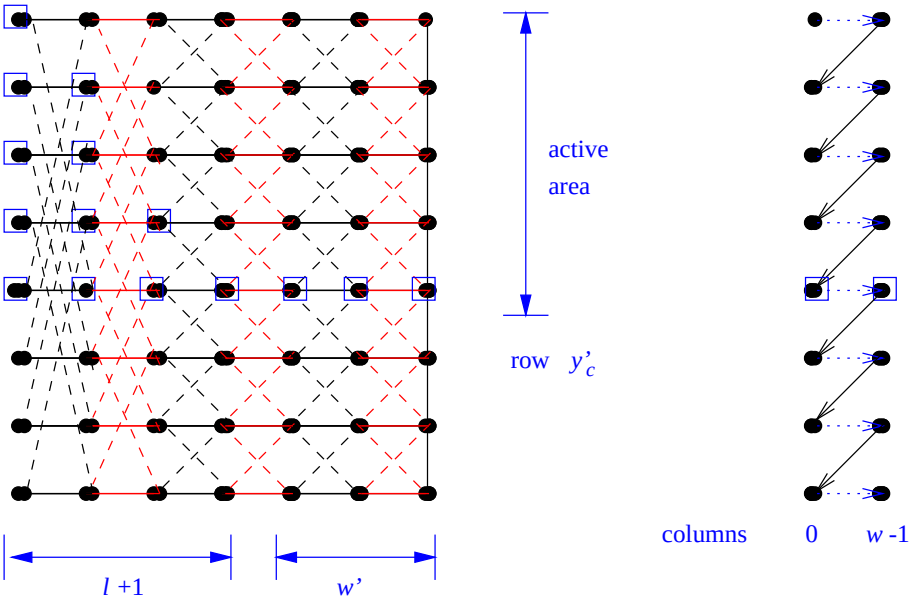


Figure 21: The network $P_{3,3}$ without the left-right comparators. The horizontal layers are drawn with solid lines, while the back-jump layers are drawn with dashed lines.



(a) (b)

Figure 22: View of $P_{3,3}$ after folding the left half (a) and the comparator connections between the columns 0 and $w-1$ (b).

5.2.1 Runtime analysis of $P_{l,w'}$

Recall that $n = 2(l+1+w') \cdot 2^l$ is the number of registers of $P_{l,w'}$.

Lemma 5.14 *For $k < w'/6$, the network $P_{l,w'}$ sorts any k-disturbed configuration in $O(w'+l)$ iterations.*

Let R, w, h, r, Y, J, M be defined as before. Let $L = (L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7)$ be the sequence of layers of $P_{l,w'}$. (i.e. $L_1 = L_3 = L_5 = L_7 = M'$ and $L_{2i} = M_i$.) Let C be the set of all comparators of $P_{l,w'}$.

It is enough to show that $P_{l,w'}$ sorts any k-disturbed configuration consisting of zeroes and ones in $O(w'+l)$ iterations. Let c be an arbitrary zero-one k-disturbed configuration of R. Let z be a number of zeroes in c. Then $y'_c = \lfloor z/w \rfloor$ is the index of the first row of registers that intersects the ones area. For $t \geq 0$, let c_t be a configuration obtained after execution of t steps of $P_{l,w'}$ on c.

Let c' be a configuration of R defined as follows.

$$c'(r) = \begin{cases} 0 & \text{if } c(r) = 0, \\ 1 + |\{p \in R \mid p < r,\ c(p) = 1\}| & \text{if } c(r) = 1 \text{ and } r = r(x,y) \text{ for } y < y'_c, \\ k+1 & \text{if } c(r) = 1 \text{ and } r = r(x,y) \text{ for } y \geq y'_c. \end{cases}$$

The configuration c has at most k displaced ones. Thus in c' the registers above the row y'_c , which contain displaced ones in c, contain the value from the range $\{1, \ldots, k\}$, every value occurring exactly once.

Let the sequences of the configurations c'_t and c''_t be defined as follows:

- $c''_0 = c'$, and
- for $t \geq 0$, c'_t is the configuration c''_t with all the values from the range $\{1, \ldots, k\}$ below the row $y'_c - 1$ replaced by the value $k+1$.
- for $t \geq 0$, $c''_{t+1} = L_{t \bmod 8}(c'_t)$.

The next claim follows directly from the definitions introduced:

Claim 5.15 *For each $t \geq 0$, the configuration c'_t has the following properties:*

1. *For each register r, $c_t(r) = 1$ if and only if $c'_t(r) > 0$.*
2. *If a register r is above the row $y'_c - 1$, then $0 \leq c'_t(r) \leq k$.*
3. *If a register r is below the row $y'_c - 1$, then $c'_t(r) = 0$ or $c'_t(r) = k+1$.*
4. *For each $i \in \{1, \ldots, k\}$, there is at most one register r such that $c'_t(r) = i$.*

We will show that all positive values of c'_t leave the region above the row $y'_c - 1$ in $O(l+w')$ iterations. By Claim 5.15 that means that for some constant d, the configuration $c_{d(l+w')}$ contains no ones above row $y'_c - 1$ and (since the network is symmetrical and we have placed no restrictions on c) $c_{d(l+w')}$ contains no zeroes below $y'_c + 1$. Thus $c_{d(l+w')}$ is at most $3w$-dirty and (by Fact 5.13 and Lemma 3.4) will be sorted in the next $O(w) = O(w'+l)$ iterations.

For $t \geq 0$, for each pair of distinct values q_1 and q_2 from the range $\{1, \ldots, k\}$ such that $q_1 < q_2$, we say that the value q_1 has been *blocked* by the value q_2 in the register r at step t if and only if there is a comparator $(r, r') \in L_{t \bmod 8}$ and $c'_t(r) = q_1$ and $c'_t(r') = q_2$. We say that the value q_1 has been *pulled back* from register r' to the register r by the value q_2 at step t if and only if there is a comparator $(r, r') \in L_{t \bmod 8}$ and $c'_t(r) = q_2$ and $c'_t(r') = q_1$. (So then we get $c'_{t+1}(r) = q_1$ and $c'_{t+1}(r') = q_2$)

Back-jump paths and stoppers. Below, we start to investigate in detail the fine structure of $P_{l,w'}$. We call the set of registers above the row y'_c an *active area*. The registers above the row y'_c are called *active registers*. We call the active registers with horizontal coordinates less than $w/2$ the *left active* registers. The remaining registers are called the *right active* registers. Let S' denote the set of active registers and let $S'_0 = S_0 \cap S'$ and $S'_1 = S_1 \cap S'$.

We call an right active register r a *back-jump starter* if and only if there is no back-jump comparator of the form (r', r). We call an active register r a *back-jump stopper* if and only if there is no back-jump comparator of the form (r, r') such that r' is in the active area. For each back-jump starter r, we define a *back-jump path of* r as the longest sequence of active registers $(r_0, \ldots, r_s)$, such that $r_0 = r$, and for $0 \leq t < s$ there is a back-jump comparator (r_t, r_{t+1}). Note that r_s is a stopper, and that each right active register is on exactly one back-jump path.

On the Fig. 21, we depict the active area for some configuration c. The back-jump stoppers are marked with the boxes. Observe that the length of each back-jump path is not greater than $w/2$. (Indeed, column index of a starter is not greater than $w-1$, the horizontal coordinate decreases by one as we go from one register of the back-jump path to the next one, and all the active registers in column $w/2$ are stoppers.)

We consider the positions of each positive value in the active area in the configurations c'_t. We call the values from the range $\{1, \ldots, k\}$ *active*. Note that each active value may disappear (be replaced by $k+1$) if it is compared with a zero from outside the active area.

Zones and the levels of registers. We partition the set S'_1 into zones $Z_{i,j}$ and $Z'_{i,j}$ defined as follows (see Fig. 23 (a)):

- for $0 \leq x \leq l-1$

  $$Z_{x,0} = \{r(w/2+x, y) \in S'_1 \mid \ r(w/2+x, y) \text{ is a stopper and } y'_c - y > 2^{l-1-x}\},$$

- for $0 \leq x \leq l-1$

  $$Z'_{x,0} = \{r(w/2+x, y) \in S'_1 \mid \ r(w/2+x, y) \text{ is a stopper and } y'_c - y \leq 2^{l-1-x}\},$$

- for $l \leq x \leq w/2 - 1$

  $$Z_{x,0} = \{r(w/2+x, y'_c - 1)\},$$

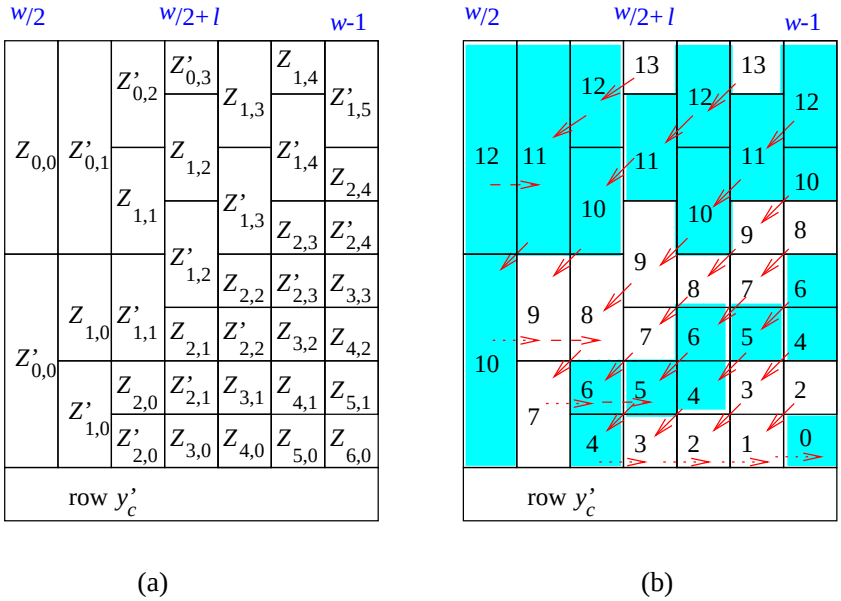


Figure 23: Partition of the right active registers into the zones (a) and the levels of the zones (b). The arrows on (b) represent the arcs of the tree of zones T.

- for $x' > 0$

  $Z_{x,x'} = \{r \in S'_1 \mid$ there is a back-jump comparator (r, r') such that $r' \in Z_{x,x'-1}\ \}$,

  and

  $Z'_{x,x'} = \{r \in S'_1 \mid$ there is a back-jump comparator (r, r') such that $r' \in Z'_{x,x'-1}\ \}$.

It follows directly from the definition that each zone is contained in a single column of registers. We call the stoppers from the zones $Z_{x,0}$ *upper stoppers* and the stoppers from the zones $Z'_{x,0}$ *lower stoppers*.

We define a *zones tree* T as a directed graph with the set of vertices $V = \{Z_{x,x'} | Z_{x,x'} \neq \emptyset\} \cup \{Z'_{x,x'} | Z'_{x,x'} \neq \emptyset\}$ and the set of arcs $E = E_1 \cup E_2$ (see arrows on Fig. 23 (b)), where

$$E_1 = \{(Z_{x,x'+1}, Z_{x,x'}) \in V \times V\} \cup \{(Z'_{x,x'+1}, Z'_{x,x'}) \in V \times V\}$$

and

$$E_2 = E_{2,1} \cup E_{2,2} \cup E_{2,3},$$

where

$$E_{2,1} = \{(Z_{x,0}, Z'_{x,1}) \in V \times V \mid 0 \leq x \leq l-1\},$$

and

$$E_{2,2} = \{(Z'_{x,0}, Z_{x+1,0}) \in V \times V \mid 0 \leq x \leq l-1\},$$

and

$$E_{2,3} = \{(Z_{x,0}, Z_{x+1,0}) \in V \times V \mid l \leq x \leq w/2\}.$$

Arcs of E_1 are called *back-jump arcs* and arcs of E_2 are called *horizontal arcs*. The arrows on Figure 23 (b) correspond to the arcs of the zones tree. The solid arrows represent E_1. The dashed arrows represent $E_{2,1}$. The dotted arrows represent $E_{2,2}$ and $E_{2,3}$.

The *root* of T is $Z_{w/2-1,0}$ (i.e. the singleton containing the last register of the active area).

For each zone $Z \in V$ we define its *level* (denoted by level(Z)) as a distance from the root in the zones tree T. For each active register $r \in S_1$, we define level(r) as the level of the zone containing r. For each active register $r \in S_0$ we define: level(r) = level(shd(r)). For each active value i, let *level of* i in configuration c'_t denote the level of active register that contains value i or zero if i does not exist in c'_t. The levels of the zones are displayed on Fig. 23 (b).

Claim 5.16 *The maximal vertex level in the tree T is $O(w' + l)$.*

Proof. Consider the path from an arbitrary vertex of T to the root of T. First we make some t_1 steps by the back-jump arcs, until we reach the first zone consisting of the stoppers. Thus $0 \leq t_1 \leq w/2$, since we can go through at most $w/2$ columns leftwards. Then, while we are in the l leftmost columns of S'_1, we need at most two steps to advance from the zone of upper stoppers to the zone of lower stoppers in the same column, and then we make three steps each time to go from the zone of lower stoppers to the zone of lower stoppers in the next column on the right side. As soon as we enter any zone of stoppers in the columns $w/2 + l, \ldots, w - 1$, we go to the next column on the right side during each single step. □

Let l'_T denote the maximal level of a (non-empty) zone in T. We partition the set of levels into *layers of levels*. Note that for $0 < x < l$, we have level($Z'_{x-1,0}$) − level($Z'_{x,0}$) = 3. Let b = level($Z'_{0,0}$) mod 3. We define the ith layer of levels as

$$\mathcal{L}_i = \{j \mid 3i + b \leq j < 3(i + 1) + b\}.$$

The zones with the levels in the odd and in the even layers have been depicted by different shades on the Fig. 23. Note that by the definition of b, the level of each zone $Z'_{x,0}$ is a minimum of some layer of levels.

For each $l \geq 0$, we define the set of registers A_l as follows:

$$A_l = \{r \in S'_1 \mid \text{level}(r) \leq l\}.$$

Thus A_l is the union of the zones with the levels not greater than l. Note also that $A_{l'_T} = S'_1$ and $A_0 = Z_{w/2-1,0}$.

The following three claims follow from the definition of the network and from the definitions of the zones and the levels of the active registers.

Claim 5.17 *If (r, r') is a comparator such that r and r' are active registers, then one of the following cases holds:*

1. (r, r') *is a left-right comparator and* $r' = \mathrm{shd}(r)$ *and hence* $\mathrm{level}(r) = \mathrm{level}(r')$.

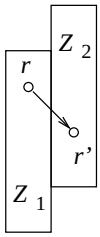
2. (r, r') *is a horizontal comparator and* $r, r' \in S'_1$, *and either* $\mathrm{level}(r') = \mathrm{level}(r) - 1$, *or* $r \in Z'_{x,0}$ *and* $r' \in Z'_{x+1,0}$ *for some* $0 \leq x \leq l-2$ *and* $\mathrm{level}(r') = \mathrm{level}(r) - 3$, *or there is a back-jump arc in* T *from the zone containing* r' *to the zone containing* r *and* $\mathrm{level}(r') = \mathrm{level}(r) + 1$.

3. (r, r') *is a horizontal comparator and* $r, r' \in S'_0$, *and either* $\mathrm{level}(r') = \mathrm{level}(r) + 1$, *or* $r \in \mathrm{shd}(Z'_{x+1,0})$ *and* $r' \in \mathrm{shd}(Z'_{x,0})$ *for some* $0 \leq x \leq l-2$ *and* $\mathrm{level}(r') = \mathrm{level}(r) + 3$, *or there is a back-jump arc in* T *from the zone containing* $\mathrm{shd}(r)$ *to the zone containing* $\mathrm{shd}(r')$ *and* $\mathrm{level}(r') = \mathrm{level}(r) - 1$.

4. (r, r') *is a horizontal comparator and* r *is in column* $w-1$ *and* r' *is in column* 0, *and* $\mathrm{level}(r') = \mathrm{level}(r) - 2$ *or* $\mathrm{level}(r') = \mathrm{level}(r)$.

5. (r, r') *is a back-jump comparator and* $r, r' \in S'_1$, *and* $\mathrm{level}(r') = \mathrm{level}(r) - 1$.

6. (r, r') *is a back-jump comparator and* $r, r' \in S'_0$, *and either* $\mathrm{level}(r') < \mathrm{level}(r)$ *or* $\mathrm{level}(r') = \mathrm{level}(r) + 1$. *The second case is only possible if the zones* Z_1 *and* Z_2 *containing respectively* $\mathrm{shd}(r)$ *and* $\mathrm{shd}(r')$ *are connected by the arc* (Z_2, Z_1) *in* T *(see the following figure).*



Claim 5.18 *If* $r \in S'_1$ *and* $\mathrm{level}(r) > 0$, *then there is a horizontal or back-jump comparator* (r, r') *such that* $r' \in S'_1$ *and either* $\mathrm{level}(r') = \mathrm{level}(r) - 1$ *or* $\mathrm{level}(r') = \mathrm{level}(r) - 3$. *The case* $\mathrm{level}(r') = \mathrm{level}(r) - 3$ *occur only when* r *is in the lower half of* $Z'_{x,0}$ *and* $r' \in Z'_{x+1,0}$, *for some* $0 \leq x \leq l-2$, *and* (r, r') *is a horizontal-comparator.*

Claim 5.19 *Let* $i \in \{0, 4\}$ *(i.e.* L_i *is a horizontal layer). For each comparator* $(r, r') \in L_i \cap (S'_1 \times S'_1)$ *such that* $\mathrm{level}(r') = \mathrm{level}(r) + 1$, *there is a back-jump comparator* $(r', r'') \in L_{i+2} \cap (S'_1 \times S'_1)$ *such that* $\mathrm{level}(r'') = \mathrm{level}(r)$.

Releasing the layers of register levels by the displaced values. For each $t \geq 0$, for $1 \leq i \leq k$, we say that a register r *is released* from the value i at step t if and only if for each $t' \geq t$, $c'_t(r) \neq i$.

To conceive the idea used in the analysis of this part of the computation consider the following simple example: Suppose we have an odd-even transposition sorting network (see Definition 3.2) with a configuration consisting of the

k positive values: $1, \ldots, k$, placed in arbitrary registers and zeroes placed in all the remaining registers. Note that at the first computation step (i.e. after applying the first layer), the first register is released from the greatest value (i.e. from k). After the second step the first and the second registers are released from the value k and thus after the third step the first register is released from $k-1$. In such a way we can define for any $t \geq 0$ and $i < k$ the set of registers that *must* be released from the value $k-i$ at step t. Note that the border of the area that must be released from the value $k-i-1$ is adjacent to the border of the area that must be released from the value $k-i$. In our network we use the subsets A_l for a construction of analogous sets.

Recall that by a *step* we mean an application of a single layer of the network to the current configuration, while by an *iteration* we mean the application of the entire sequence of layers of a periodic network.

Recall also that each active value may disappear, thus releasing all active registers. For the simplicity, we skip this case in the proofs of the following claims.

In the following we assume that $w' \geq 6k$.

The aim of the *first phase* of the computation is to move each positive value i in the active region into

$$A_{6(k-i)+6} \cup \mathrm{shd}(A_{6(k-i)+6}) \subseteq A_{w'} \cup \mathrm{shd}(A_{w'}).$$

Claim 5.20 *Let $t \geq 0$ and $j > 0$ be integers. For each active value i, if all active registers outside $A_{\max \mathcal{L}_j} \cup \mathrm{shd}(A_{\min \mathcal{L}_j})$ are released from the values greater than i at step t, and i is inside $A_{\min \mathcal{L}_{j+1}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+1}})$ in configuration c'_t, then all active registers outside $A_{\max \mathcal{L}_{j+1}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+1}})$ are released from i at step t.*

Proof. We have to show that if the values greater than i remain in the area $A_{\max \mathcal{L}_j} \cup \mathrm{shd}(A_{\min \mathcal{L}_j})$, and the value i is inside $A_{\min \mathcal{L}_{j+1}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+1}})$, then i will never leave $A_{\max \mathcal{L}_{j+1}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+1}})$.

Assume that the values greater than i have released all active registers that are outside $A_{\max \mathcal{L}_j} \cup \mathrm{shd}(A_{\min \mathcal{L}_j})$.

Fact 5.21 *If the value i is in the $\mathrm{shd}(A_{\min \mathcal{L}_{j+1}})$, then as soon as it leaves $\mathrm{shd}(A_{\max \mathcal{L}_j})$ it must enter S_1 in at most one more (left-right) step and it can enter S_1 only inside $A_{\min \mathcal{L}_{j+1}}$.*

The first part of the fact is implied by the fact that all active registers outside $A_{\max \mathcal{L}j}$ are released from the values greater than i.

The second part of the fact follows from the definition of $\mathcal{L}_j$ and of the levels of the zones: The value i can enter $S' \setminus \mathrm{shd}(A_{\max \mathcal{L}_j})$ only by being pulled back by some greater value (with a horizontal comparator between columns $w-1$ and 0) directly to $A_{\max \mathcal{L}_j} \subseteq A_{\min \mathcal{L}_{j+1}}$ or by going forward through the comparators that have the first endpoint in $\mathrm{shd}(A_{\max \mathcal{L}_j})$ and the second endpoint in $S' \setminus \mathrm{shd}(A_{\max \mathcal{L}_j})$. In the second case i either enters $A_{\max \mathcal{L}_j} \subseteq A_{\min \mathcal{L}_{j+1}}$ through some left-right comparator or enters $\mathrm{shd}(A_{\min \mathcal{L}_{j+1}}) \setminus \mathrm{shd}(A_{\max \mathcal{L}_j})$ through some

comparator contained in $S'_0 \times S'_0$. (Note that by the Claim 5.17 (6) the back-jump comparator can increase the level of the register containing i by at most one, and by the definition of $\mathcal{L}_j$, the horizontal comparator from $S'_0 \times S'_0$ can not move i directly outside $\mathrm{shd}(A_{\min \mathcal{L}_{j+1}})$.)

Fact 5.22 *If the value i is in the $A_{\min \mathcal{L}_{j+1}}$, then it cannot be moved outside $A_{\min \mathcal{L}_{j+1}+1} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+1}})$.*

The comparators that can move the value i from $A_{\min \mathcal{L}_{j+1}}$ to $S'_1 \setminus A_{\min \mathcal{L}_{j+1}}$ must have the first endpoint in $A_{\min \mathcal{L}_{j+1}}$ and the second endpoint outside $A_{\min \mathcal{L}_{j+1}}$ in S'_1. The only such comparators are the horizontal comparators with the first register in $A_{\min \mathcal{L}_{j+1}}$ and the second register in $A_{\min \mathcal{L}_{j+1}+1} \setminus A_{\min \mathcal{L}_{j+1}}$. But, by the Claim 5.19, as soon as i is moved to $A_{\min \mathcal{L}_{j+1}+1} \setminus A_{\min \mathcal{L}_{j+1}}$, it is moved back to the zone with the level $\min \mathcal{L}_{j+1}$ by the following back-jump step, since the zones with the level $\min \mathcal{L}_{j+1}$ are released from the values greater than i. If the value i leaves S_1, then it must be shifted by a horizontal comparator from the column $w-1$ to the column 0 to the shadow of the zone with not greater level or be pulled back by a greater value somewhere inside $\mathrm{shd}(A_{\min \mathcal{L}_j})$ through some left-right comparator. □

Claim 5.23 *Let $t \geq 0$ and $j > 0$ be integers. For each active value i, if all active registers outside $A_{\max \mathcal{L}_j} \cup \mathrm{shd}(A_{\min \mathcal{L}_j})$ are released from the values greater than i at step $8t$ (i.e. after iteration t), and the active registers outside $A_{\max \mathcal{L}_{j+2}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+2}})$ are released from i at step $8t$, then after iteration $t+6$ (i.e. after step $8(t+6)$), the active registers outside $A_{\max \mathcal{L}_{j+1}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+1}})$. are released from the value i.*

Proof. Assume that after iteration t we have all active registers outside $A = A_{\max \mathcal{L}_j} \cup \mathrm{shd}(A_{\min \mathcal{L}_j})$ released from the values greater than i and i has released all active registers outside $A_{\max \mathcal{L}_{j+2}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+2}})$

We show that within the next 6 iterations the value i visits some register from $A' = (A_{\min \mathcal{L}_{j+1}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+1}}))$. By Claim 5.20, as soon as i enters A' it releases all active registers that are outside $A_{\max \mathcal{L}_{j+1}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+1}})$.

If after iteration t, the value i is inside $(A_{\max \mathcal{L}_{j+2}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+2}})) \setminus A'$, then in at most the next 2 steps it must either:

- enter A', or
- be moved by the first left-right layer to $A_{\max \mathcal{L}_{j+2}} \setminus A_{\min \mathcal{L}_{j+1}}$, since it can not be blocked by any greater value outside A'.

After that either:

- i starts being moved by the back-jump comparators in S_1, or
- (if i is in the column $w-1$) i can be moved by the next horizontal layer to $\mathrm{shd}(A_{\max \mathcal{L}_{j+2}})$ and then (if i is still outside A') back to $A_{\max \mathcal{L}_{j+2}}$ by the following left-right layer, or

- i can be moved by the next horizontal layer to some register in S'_1 with the level not greater than $\max \mathcal{L}_{j+2}$ (since the registers with the levels greater than $\max \mathcal{L}_{j+2}$ are released from i).

After that (if i is still in $A_{\max \mathcal{L}_{j+2}} \setminus A_{\min \mathcal{L}_{j+1}}$) the value i is moved only by the back-jump comparators inside S_1 until i enters $A_{\min \mathcal{L}_{j+1}}$ or reaches a stopper. (Each such back-jump step happens every four computation steps and decreases level of i by at least 1.)

If i is still inside $A_{\max \mathcal{L}_{j+2}} \setminus A_{\min \mathcal{L}_{j+1}}$, then in the next iteration it starts moving through either horizontal or back-jump comparators in S'_1 until it enters $A_{\min \mathcal{L}_{j+1}}$. Indeed, each time i is moved in this phase, the level of i is decreased either by 1 (if i is moved by a back-jump or horizontal comparator from a zone Z_1 to Z_2 such that (Z_1, Z_2) is an arc in the tree of zones T) or by 3 (if, for some $x < l-1$, i is moved from the lower half of the zone $Z'_{x,0}$ to $Z'_{x+1,0}$ by a horizontal comparator). Note that if $Z'_{x+1,0} \subseteq A$, then $Z'_{x,0} \subseteq A_{\min \mathcal{L}_{j+1}}$, since the levels of zones of lower stoppers are minimal within their layers of levels. Thus, i cannot be blocked by a greater active value unless it is in A'.

We have shown that latest of all at the second iteration following the iteration t, the level of i starts being decreased by at least 1 each iteration. Since the initial level is not greater than $\max \mathcal{L}_{j+2}$ and $\max \mathcal{L}_{j+2} - \min \mathcal{L}_{j+1} = 5$, the total number of the iterations needed to move i to $A_{\min \mathcal{L}_{j+1}}$ is not greater than 6. □

For $t \geq 0$, for $1 \leq i \leq k$ we define the the *tth level limit for* i, denoted $l_{t,i}$ as follows:

$$l_{t,i} = \max \mathcal{L}_{\max\{2(k-i), l'_T - t + 2(k-i)\}}.$$

Claim 5.24 *Let $t \geq 0$. For each i, $1 \leq i \leq k$, the active registers outside $A_{l_{t,i}} \cup \mathrm{shd}(A_{l_{t,i}-2})$ are released from the value i after the iteration $6 \cdot t$.*

Proof. Let $1 \leq i \leq k$. For $t = 0$, we have $A_{l_{0,i}} \cup \mathrm{shd}(A_{l_{0,i}-2}) = A_{\max \mathcal{L}_{l'_T+2(k-i)}} \cup \mathrm{shd}(A_{\max \mathcal{L}_{l'_T+2(k-i)}-2}) = A_{3(l'_T+2(k-i))+b+2} \cup \mathrm{shd}(A_{3(l'_T+2(k-i))+b}) = S'_1 \cup S'_0 = S'$. Thus we have proven the case $t = 0$ for all the values i, $1 \leq i \leq k$.

For the case $i = k$, consider the behavior of the value k in the active area. After the first iteration k must be placed in S_1 outside the column $w-1$ (if it is above the row $y'_c - 1$) and it remains there, since there is no greater value in the active area that could pull it back to S_0. Then at the next back-jump step with the comparators that have the first registers in the zone containing k, the value k starts to travel through the zones of S_1 to the root of T. During each step of the travel, the value k either is moved to the zone connected with its current zone by an arc or if it is in the lower half of some zone $Z'_{x,0}$, for $0 \leq x \leq l-2$, to the zone $Z'_{x+1,0}$. At least one step of the travel is performed during each iteration.

Thus after the first iteration all the active registers that are outside $A_{3l'_T-1} = A_{l'_T}$ are released from the value k. After the tth iteration (and hence after the iteration $6t$) all the active registers that are outside $A_{\max\{0,3l'_T-t\}}$ are released

from the value k. Since

$$A_{\max\{0,3l'_T-t\}} \subseteq A_{l_{t,k}} \cup \mathrm{shd}(A_{l_{t,k}-2})$$

the case $i = k$ has been proven.

Let $1 \leq i < k$ and $t > 1$. By the induction hypothesis, all active registers outside $A_{l_{t-1,i}} \cup \mathrm{shd}(A_{l_{t-1,i}-2})$ are released from i after $6(t-1)$ iterations and all the active registers outside $A_{l_{t-1,i+1}} \cup \mathrm{shd}(A_{l_{t-1,i+1}-2})$ are released from the values greater than i after $6(t-1)$ iterations. We have $l_{t-1,i} = \max \mathcal{L}_{\max\{2(k-i),l'_T-t+1+2(k-i)\}}$ and $l_{t-1,i+1} = \max \mathcal{L}_{\max\{2(k-i-1),l'_T-t+1+2(k-i-1)\}}$. Let $j = \max\{2(k-i-1), l'_T-t+1+2(k-i-1)\}$. Thus all active registers outside $A_{\max \mathcal{L}_j} \cup \mathrm{shd}(A_{\min \mathcal{L}_j})$ are released from the values greater than i at step t, and active registers outside $A_{\max \mathcal{L}_{j+2}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+2}})$ are released from i at step t, and by the Claim 5.23, the active registers outside $A_{\max \mathcal{L}_{j+1}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{j+1}})$ are released from the value i within the next 6 iterations. □

Corollary 5.25 *For each i, $1 \leq i \leq k$, the active registers that are outside $A_{\max \mathcal{L}_{2(k-i)}} \cup \mathrm{shd}(A_{\min \mathcal{L}_{2(k-i)}})$ are released from the value i after $6l'_T$ iterations.*

Note that $\max \mathcal{L}_{2(k-i)} = 6(k-i) + b + 2$. Thus after $6l'_T$ iterations of computation all positive values will remain in the zones with levels not greater than $6k - 4 + b$ and their shadows. Since $b \leq 2$ and $w' \geq 6k$, all those zones are the singletons in $A_{w'} \cup \mathrm{shd}(A_{w'})$ in the rightmost w' columns (in the folded version of $P_{l,w'}$). The connections between the registers in this part of the network have a very regular structure. We use this regularity to show that all positive values will flow into the row $y'_c - 1$ in the next $O(w')$ iterations.

The following claim is a collection of some properties of $(A_{w'} \cup \mathrm{shd}(A_{w'}))$-restriction of the network, useful in the analysis of the next two phases of the network computation.

Claim 5.26 *1.* $\mathrm{shd}(A_{w'})$ *is contained in the columns* $0, \ldots, w'-1$,

2. $A_{w'}$ *is contained in the columns* $w-w', \ldots, w-1$,
3. $A_{w'} \cup \mathrm{shd}(A_{w'})$ *is contained in the rows* $y'_c-1, \ldots, y'_c-1-\max\{y \mid w' \geq 2y\}$
4. *For* $0 \leq i \leq \max\{y \mid w' \geq 2y\}$, *for* $0 \leq j \leq w'-2i$, $\mathrm{level}(r(w-1-j, y'_c - 1 - i)) = \mathrm{level}(r(j, y'_c - 1 - i)) = j + 2i$.
5. *For each* $r(x,y) \in \mathrm{shd}(A_{w'})$, *there is a left-right comparator* $(r(x,y), \mathrm{shd}(r(x,y)))$ *and* $\mathrm{level}(\mathrm{shd}(r(x,y))) = \mathrm{level}(r(x,y))$.
6. *For each* $r(x,y) \in A_{w'}$ *such that* $y < y'_c - 1$, *there is a back-jump comparator* $(r(x,y), r(x-1, y+1))$ *and* $\mathrm{level}(r(x-1,y+1)) = \mathrm{level}(r(x,y)) - 1$.
7. *For each* $r(x,y) \in A_{w'}$ *such that* $x < w-1$, *there is a horizontal comparator* $(r(x,y), r(x+1,y))$ *and* $\mathrm{level}(r(x+1,y)) = \mathrm{level}(r(x,y)) - 1$.
8. *For each comparator* $(r, r') \in A_{w'} \times A_{w'}$, $\mathrm{level}(r') = \mathrm{level}(r) + 1$.

9. *There is a comparator* $(r(w-1, y'_c - 2), r(0, y'_c - 1))$. *(The entrance to the shadow of last row of* A_k*.)*

10. *For each* $0 \leq i < k$*, there is a comparator* $(r(i, y'_c - 1), r(i+1, y'_c - 1))$. *(The existence of the Hamiltonian path of comparators in the shadow of the last row of* A_k*.)*

Proof. The properties listed follow directly from the definition of the levels of registers. □

We start the *second* phase of computation with a configuration where each positive value i, $1 \leq i \leq k$ is inside $A_{6(k-i)+b+2}$ or its shadow, or does not exist, and our aim is to obtain a configuration such that each i, $1 \leq i \leq k$ is inside A_{k-i}. The second phase is very similar to the first phase, but the part of a network occupied by the active values in the second phase has a very simple structure.

Claim 5.27 *Assume that we start some iteration of the second phase with a configuration such that for some* j, $0 \leq j \leq w'$, *for each* i, $1 \leq i \leq k$, *the active registers outside* $A_{\max\{k-i,j-2i\}} \cup \mathrm{shd}(A_{\max\{k-i,j-2i\}})$ *are released from the value* i. *Then after the next iteration for each* i, $1 \leq i \leq k$, *the active registers outside* $A_{\max\{k-i,j-1-2i\}} \cup \mathrm{shd}(A_{\max\{k-i,j-1-2i\}})$ *are released from* i.

Proof. For the value k the claim is obvious, since k is never blocked in the active region. Consider any $i < k$. The registers that are outside $A_{\max\{k-i-1,j-2i-2\}} \cup \mathrm{shd}(A_{\max\{k-i-1,j-2i-2\}})$ are released from all the values greater than i, and i must be inside $A_{\max\{k-i,j-2i\}} \cup \mathrm{shd}(A_{\max\{k-i,j-2i\}})$. If i is outside $A_{\max\{k-i,j-2i-2\}} \cup \mathrm{shd}(A_{\max\{k-i,j-2i-2\}})$, then (after last left-right step of the previous iteration) i must be placed in some register with the level $\max\{k-i, j-2i\}$ or $\max\{k-i, j-2i-1\}$ in S'_1 and by the Claim 5.26 (points 6 and 7) there is a comparator that moves it to the register in the $A_{\max\{k-i,j-2i-1\}}$ unless i is already there. □

Corollary 5.28 *After* $O(w')$ *iterations of the second phase, each active value* i *has released active registers that are outside* $A_{k-i} \cup \mathrm{shd}(A_{k-i})$.

Proof. The corollary follows from Claim 5.27. □

Note that after the last left-right step of the last iteration the value k (if still exists) must be in the only register of A_0. The value $k-1$ can be in one of the four registers of $A_1 \cup \mathrm{shd}(A_1)$. We add one more iteration to the second phase to ensure that $k-1$ is moved to A_1. (We will use this in the proof of Claim 5.32.)

Final smoothing of displaced elements. The *third* phase we start in a configuration, where each positive active value i is inside $A_{k-i} \cup \mathrm{shd}(A_{k-i})$ or does not exist. (Moreover, k and $k-1$ are in A_0 and A_1 respectively.)

We modify slightly the definition of the configurations c'_t in the third phase of the computation: If t is the number of the computation step in the third phase

of the computation, then c'_t is obtained from the configuration c''_t by replacing all positive values below the active area *and in the last row of* A_k with the value $k+1$.

Our aim is to obtain all active values inside the row $y'_c - 1$. Let B denote the last row of A_k and let $B' = \mathrm{shd}(B)$ (i.e. B and B' are contained in the row $y'_c - 1$). All active values are inside $A = A_k \cup \mathrm{shd}(A_k)$. The third phase will move all active values to the lowest row of A. There are many ways the active values can enter the last row of A, however we concentrate only on the horizontal comparator $(r(y'_c - 2, w-1), r(y'_c - 1, 0))$ (mentioned as the entrance to the shadow of the last row in point 9 of Claim 5.26). To each register r in A we assign a label $q(r)$ that may increase during the computation. The label $q(r)$ is either integer value or an integer value plus 0.5, and $k - q(r)$ is an upper bound on the positive value that can still appear in r. Since all positive values that enter B are immediately replaced by $k+1$, the registers of B can have label -1. The values k and $k-1$ are in B already before the third phase. Thus all the registers in B' are released from k and $k-1$ and can be initially labeled by 1.5. The registers in $A \setminus (B \cup B')$ have the initial labels equal to their levels. If for some $r \in A$ there is a back-jump or horizontal comparator inside A_k or the entrance to B', of the form (r, r') or $(\mathrm{shd}(r), r')$ such that $q(r) - q(r') = 0.5$, then either:

- the value $k - q(r)$ is an integer and the register r' is released from $k - q(r) + 1$ (thus the value $k - q(r)$ can move from r to r' in at most single iteration and we can then increase $q(r)$ by 0.5.), or
- the value $k - q(r)$ is not an integer and we can increase $q(r)$ to the next greater integer (by adding the value 0.5) without destroying the upper bound on the positive values that can be in r.

In a similar way we can increase the labels of registers in B' with the use of horizontal comparators contained in B'.

Here is a more formal definition of the labels. We assign the labels $q_t(r)$ to the registers r of $A_k \cup \mathrm{shd}(A_k)$ as follows:

- For $r \in B$, for all $t \geq 0$:
$$q_t(r) = -1.$$
- For $r \in B'$:
$$q_0(r) = 1.5.$$
- For $r \notin B \cup B'$:
$$q_0(r) = \mathrm{level}(r).$$
- For $t > 0$:
$$q_t(r(w-1, y'_c - 2)) = q_{t-1}(r(w-1, y'_c - 2)) + 0.5$$
and
$$q_t(r(0, y'_c - 2)) = q_{t-1}(r(0, y'_c - 2)) + 0.5$$

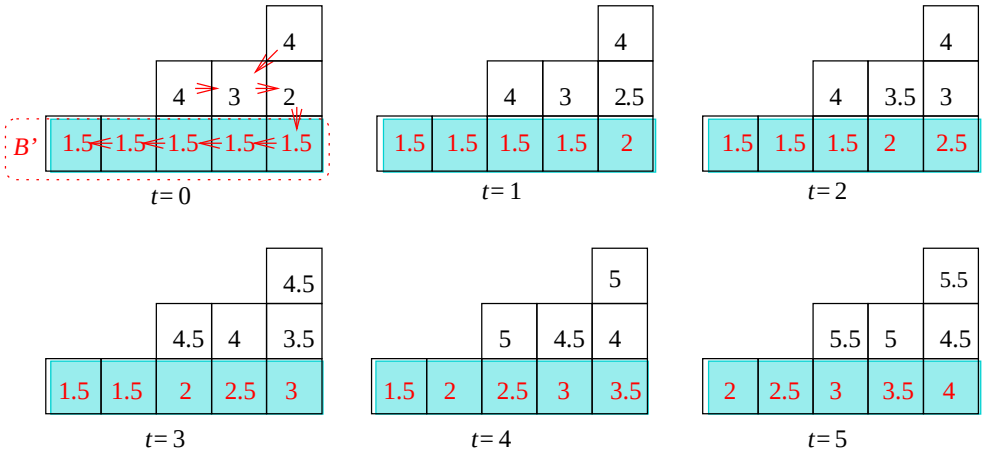


Figure 24: The labeling of the registers of $(A_k \setminus B) \cup B'$ of the folded $P_{l,w'}$, where $k = 4$, for the subsequent values of t.

and

$$q_t(r(0, y'_c - 1)) = q_{t-1}(r(0, y'_c - 1)) + 0.5.$$

- For $t > 0$, for $r(x, y) \in B'$, $x > 0$:

$$q_t(r(x, y'_c - 1)) = q_{t-1}(r(x - 1, y'_c - 1)).$$

- For $t > 0$, and $r \in A_k \setminus B$:
  if there is a horizontal or back-jump comparator (r, r') such that $r' \in A_k \setminus B$ and $q_{t-1}(r') = q_{t-1}(r) - 0.5$, then

$$q_t(r) = q_{t-1}(r) + 0.5$$

and

$$q_t(\mathrm{shd}(r)) = q_{t-1}(\mathrm{shd}(r)) + 0.5$$

else

$$q_t(r) = q_{t-1}(r)$$

and

$$q_t(\mathrm{shd}(r)) = q_{t-1}(\mathrm{shd}(r))$$

For $t \geq 0$, we define the set $Q_{t,i}$ as follows:

$$Q_{t,i} = \{r \in A_k \cup \mathrm{shd}(A_k) \mid q_t(r) \leq i\}.$$

Claim 5.29 *For each $t \geq 0$, for each $i > 2$, if $Q_{t+1,i} \setminus (B \cup B') \neq Q_{t,i} \setminus (B \cup B')$, then either there exists a register $r \in A_k$ above the row $y'_c - 1$ such that $q_t(r) = i - 0.5$ or the only registers with the label q_t equal to i are $r(w - 1, y'_c - 2)$ and* $\mathrm{shd}(r(w - 1, y'_c - 2))$.

Proof. The claim follows from the definition of the labels. □

Claim 5.30 *For each $t > 0$, if for some integer i there is a register above the row $y'_c - 1$ with the label q_t equal to $i + 0.5$, then each register above the row $y'_c - 1$ with the label q_t equal to $i + 1$ is either the first register or the shadow of the first register of the horizontal or back-jump comparator contained in $S'_1 \times S'_1$ with the second register having the label q_t equal to $i + 0.5$.*

Proof. The claim follows from the regular structure of the comparator connections in A_k and from the definition of the labels q_t. All the registers with the same level in A_k above the row $y'_c - 1$ must have the same label q_t and the registers with the greater levels must have greater values of the labels. Thus if the registers above the row $y'_c - 1$ with the level $j < k$ have the label q_t equal to $i + 0.5$, then the registers above the row $y'_c - 1$ with the level $j + 1$ must have the label q_t equal to $i + 1$ and there are no other registers above the row $y'_c - 1$ with the same label. □

Claim 5.31 *The set $Q_{2k,k}$ is contained in the row $y'_c - 1$.*

Proof. Consider the area above the row $y'_c - 1$. We may treat the fraction "0.5" as the signal that is emitted every step from the pair of registers $r(w - 1, y_c - 2)$, $\mathrm{shd}(r(w - 1, y_c - 2))$, and is broadcast to other registers in A_k above the row $y'_c - 1$ and their shadows by the horizontal and back-jump comparators contained in S'_1. (The comparator (r, r') broadcasts the signal from r' to r. The arrows on the Fig. 24 denote the comparators used for broadcasting.) Once the register receives the "0.5" signal, it starts the process of increasing its label by 0.5 every step. Once the "0.5" signal reaches the registers with level k (i.e. after $k - 1$ steps), the area of registers with the labels not greater than k starts shrinking. During each step all the labels k are replaced by the labels $k + 0.5$ and the labels k are placed on the registers that had the labels $k - 0.5$ and are "one comparator closer" to $r(w - 1, y_c - 2)$ or $\mathrm{shd}(r(w - 1, y_c - 2))$. □

Claim 5.32 *After t iterations of the third phase, each active value i, has released all active registers outside $Q_{t,k-i}$.*

Proof.

The values k and $k-1$ are in the registers $r(w-1, y'_c-1)$ and $r(w-2, y'_c-1)$ after the second phase. (Recall that we have added one more iteration to the second phase, to ensure that $k - 1$ is also in A_1.)

For $t = 0$, for $2 \leq i \leq k$, each register with the level i has the label q_0 not greater than i, thus after the second phase the value $k - i$ must be in $Q_{0,i} \supseteq A_i \cup \mathrm{shd}(A_i)$.

Consider the value $k-2$. After the last left-right step of the second phase it must be either in $r(w - 3, y'_c - 1)$ or in $\mathrm{shd}(\{r(w - 1, y'_c - 1), r(w - 2, y'_c - 1)\})$ (or in $\{r(w - 1, y'_c - 1), r(w - 2, y'_c - 1)\}$ if some of the greater values do not exist) or in the register $r(w - 1, y'_c - 2)$. In the last case, the first horizontal step containing the comparator $(r(w - 1, y'_c - 2), r(0, y'_c - 1))$ moves $k - 2$ to

the shadow of the last row of A_k (i.e to B'). As soon as $k-2$ is in B', it is moved unblocked by the subsequent horizontal steps until it is in the shadow of some register of B with the value 0, and then by the left-right step is moved to B and replaced by $k+1$. (Note that the length of B and B' is k and there are at most k positive values in $B \cup B'$ and hence $k-2$ must meet some zero in B while it is in B'.) Anyway $k-2$ will never be pulled back out of $Q_{t,2}$ after the tth iteration.

Let $t \geq 0$. (If $t = 0$, then the iteration t is the last iteration of the second phase.) Consider the placement of arbitrary positive value i, $1 \leq i \leq k-3$ after the iteration t.

If i is inside $Q_{t,k-i} \setminus (B \cup B' \cup Q_{t+1,k-i})$, then i is (after the last left-right step of the iteration t) in S'_1 in some register r above the row $y'_c - 1$ such that $q_t(r) = k-i$. There are no comparators $(r, r') \in S'_1 \times S'_1$ such that $q_t(r) < q_t(r')$. If $r \neq r(w-1, y'_c - 2)$, then by the Claims 5.29 and 5.30 there is a back-jump or horizontal comparator (r, r') such that $q_t(r') = q_t(r) - 0.5$, thus the value $k-i$ must be moved in the iteration $t+1$ to some register with the label q_t not greater than $k-i-0.5$. If $r = r(w-1, y'_c - 2)$, then the horizontal comparator $(r(w-1, y'_c - 2), r(0, y'_c - 1))$ moves $k-i$ in the iteration $t+1$ to B'.

If i is inside $Q_{t,k-i} \cap B' \setminus Q_{t+1,k-i}$, then in the iteration $t+1$ either i will be moved to the next register of B' or it will be moved to B and replaced by $k+1$.

The following fact ensures that the value i will not leave $Q_{t+1,k-i}$ during the iteration $t+1$ and later.

Fact 5.33 *For each comparator (r, r') such that $q_t(r) < q_t(r')$, we have $r \in \mathrm{shd}(A_k \setminus B)$ and $r' \in \mathrm{shd}(A_k \setminus B)$, and (r, r') is a horizontal comparator and $q_{t+1}(r') = q_t(r) + 1$.*

The fact that $r \in \mathrm{shd}(A_k \setminus B)$ and $r' \in \mathrm{shd}(A_k \setminus B)$ and that (r, r') is a horizontal comparator follows from the definition of q_t and from the structure of A_k-restriction of the network: If $q_t(r) < q_t(r')$ then (r, r') must be above $B \cup B'$, since for each t the labels in $B' \cup B$ are less than any labels in $A_k \cup \mathrm{shd}(A_k) \setminus (B' \cup B)$. On the other hand if $r, r' \in A_k \cup \mathrm{shd}(A_k) \setminus (B' \cup B)$ and $q_t(r) < q_t(r')$ then $\mathrm{level}(r) < \mathrm{level}(r')$. The only comparators (r, r') inside $A_k \cup \mathrm{shd}(A_k)$ such that $\mathrm{level}(r) < \mathrm{level}(r')$ are the horizontal comparators inside $\mathrm{shd}(A_k)$.

Let us show that $q_{t+1}(r') = q_t(r) + 1$. We have either $q_t(r') - q_t(r) = 1$ or $q_t(r') - q_t(r) = 0.5$. In the first case, $q_{t+1}(r') - q_t(r) = q_t(r') - q_t(r) = 1$. (Note that in this case the second endpoint of the back-jump comparator starting in $\mathrm{shd}(r')$ has also the label q_t less than $q_t(r') - 0.5$ and the label of r' remains unchanged.) In the second case, $q_{t+1}(r') - q_t(r) = (q_t(r') + 0.5) - q_t(r) = 1$.

If during the iteration $t+1$ the value i enters $Q_{t+1,k-i} \setminus Q_{t,k-i-1}$, then after the subsequent left-right step it must be placed in $Q_{t+1,k-i} \cap S'_1$. Thus i can never leave $Q_{t+1,k-i}$ once it have entered it, since there are no comparators in S'_1 that can move i to the register with the greater level. By the Fact 5.33, the value $k-i$ cannot go directly in single step from $Q_{t,k-i-1}$ to any register outside $Q_{t+1,k-i}$. □

Corollary 5.34 *After $2k$ iterations of the third phase all the positive values in the active area are in the row $y'_c - 1$.*

Proof. The corollary follows from the Claims 5.31 and 5.32. □

We have shown that for $w' \leq 6k$ the network $P_{l,w'}$ moves in at most $O(l + w' + k)$ iterations all the displaced ones of the k-disturbed input to the rows $y'_c - 1$ and y'_c and (by the symmetry of $P_{l,w'}$) all the displayed zeroes to the rows y'_c and $y'_c + 1$. Such a configuration is at most $3w$-dirty and can be sorted (by Lemma 3.4) in the $O(w)$ iterations of the last *fourth* phase.

For arbitrary n and k such that $k < n/6$, we can use the $\{0, \ldots, n\}$-restriction of the network $P_{l,6k}$, where $l = \min\{m \mid 2^m(m + 1 + 6k) \geq n\}$ for sorting the k-disturbed sequence of length n in $O(l + k)$ iterations. Note that l is $O(\log n)$, so the construction fulfills the properties stated.

6 Bibliography

[1] M. Ajtai, J. Komlós and E. Szemerédi. Sorting in $c \log n$ parallel steps. Combinatorica, Vol. 3, pages 1–19, 1983.

[2] M. Ajtai, J. Komlós and E. Szemerédi. Halvers and expanders. Proceedings of the 33rd IEEE-FOCS, pages 686–692, 1992.

[3] K. E. Batcher. Sorting networks and their applications. Proceedings of 32nd AFIPS, pages 307–314, 1968.

[4] Nicolas G. de Bruijn. Sorting by means of swapping. Discrete Mathematics, Vol. 9, pages 333–339, 1974.

[5] M. Dowd, Y. Perl, M. Saks, and L. Rudolph. The periodic balanced sorting network. Journal of the ACM, Vol. 36, pages 738–757, 1989.

[6] M. Kik. Correction network. Manuscript, 1997.

[7] M. Kik, M. Kutyłowski and M. Piotrów. Correction networks. Proceedings of the IEEE-ICPP, pages 40-47, 1999.

[8] M. Kik, M. Kutyłowski and G. Stachowiak. Periodic constant depth sorting network. Proceedings of the 11th STACS, pages 201–212, 1994.

[9] M. Kutyłowski, K. Loryś and B. Oesterdiekhoff. Periodic merging networks. Proceedings of the 7th ISAAC, pages 336–345, 1996.

[10] M. Kutyłowski, K. Loryś, B. Oesterdiekhoff, and R. Wanka. Fast and feasible periodic sorting networks. Proceedings of the 55th IEEE-FOCS, 1994. Full version to appear in the Journal of ACM.

[11] D. E. Knuth. The art of Computer Programming. Volume 3: Sorting and Searching. Addison-Wesley, 1973.

[12] J. G. Krammer. Lösung von Datentransportproblemen in integrierten Schaltungen. Ph.D. Dissertation, Technical University Munich, 1991.

[13] F. T. Leighton. Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes. Morgan Kaufmann, San Mateo, 1992.

[14] I. D. Scherson, S. Sen and A. Shamir. Shear-sort: A true two-dimensional sorting technique for VLSI networks, Proceedings of the IEEE-ICPP, pages 903–908. 1986.

[15] M. Schimmler, C. Starke. A Correction Network for N-Sorters. SIAM Journal on Computing, Vol. 6, No. 6, 1989.

[16] U. Schwiegelshohn. A short-periodic two-dimensional systolic sorting algorithm. IEEE International Conference on Systolic Arrays, pages 257–264, 1988.

[17] A. C. Yao. Bounds on selection networks. SIAM Journal on Computing, Vol. 9, No. 3, 1980.

A The proof of Lemma 3.21

This appendix contains the proof of Lemma 3.21 invented by Grzegorz Stachowiak.

Proof. Let $x = (m)^k(0)^{l-k}$, where $1 \leq k \leq l-1$. (The cases $k = 0$ and $k = l$ are trivial.) For $t \geq 0$ let $x_t = \mathcal{V}^t_{\varepsilon,m}(x)$. For $t \geq 0$ let $b_t \in \{0,1\}^l$ be defined as follows: $b_t = \mathcal{V}^t_{0,1}((1)^k(0)^{l-k})$. Note that for even t, b_t is the output of the t-th iteration of the l-odd-even transposition network applied to the vector $(1)^k(0)^{l-k}$, and b_{t+1} is the result of the application of the first layer of the l-odd-even transposition network to the vector b_t. By Lemma 3.3, for $t > l$, the sequences b_t are sorted.

Claim A.1 *1. If $t \geq l$, then $b_t = (0)^{l-k}(1)^k$.*

2. If $0 < t < l$, then $b_t = (0)^{\max\{0,l-k-(l-t)\}}d(1)^{\max\{0,k-(l-t)\}}$ where d is some zero-one subsequence.

The first part of the claim follows from the fact that b_t are sorted for $t \geq l$. The second part follows from the fact that b_l is sorted and that in a single layer of the odd-even transposition network the number of register containing the leftmost one can be increased by at most one and the number of register containing the rightmost zero can be decreased by at most one. □

For some $s > 0$, let $\gamma_1, \ldots, \gamma_s \geq 0$ be a sequence of coefficients such that $\sum_{i=1}^s \gamma_i = 1$. Then the vector $\gamma_1 v_1 + \ldots + \gamma_s v_s$ is a *convex combination* of the vectors $v_1, \ldots, v_s$.

Claim A.2 *Let $s > 0$, let $\gamma_1, \ldots, \gamma_s \geq 0$, such that $\sum_{i=1}^s \gamma_i = 1$, and let $v_1, \ldots, v_s \in [0,m]^l$. Then*

$$\sum_{i=1}^s \gamma_i \mathcal{N}_{\varepsilon,m}(v_i) \preceq \mathcal{N}_{\varepsilon,m}(\sum_{i=1}^s \gamma_i v_i)$$

and

$$\sum_{i=1}^s \gamma_i \mathcal{P}_{\varepsilon,m}(v_i) \preceq \mathcal{P}_{\varepsilon,m}(\sum_{i=1}^s \gamma_i v_i).$$

Proof. For even j or $j = l$, we have

$$\mathrm{hd}_j(\mathcal{N}_{\varepsilon,m}(\sum_{i=1}^s \gamma_i v_i)) = \mathrm{hd}_j(\sum_{i=1}^s \gamma_i v_i) = \mathrm{hd}_j(\sum_{i=1}^s \gamma_i \mathcal{N}_{\varepsilon,m}(v_i)).$$

For odd $j < l$,

$$\mathrm{hd}_j(\mathcal{N}_{\varepsilon,m}(\sum_{i=1}^s \gamma_i v_i)) = \mathrm{hd}_{j-1}(\mathcal{N}_{\varepsilon,m}(\sum_{i=1}^s \gamma_i v_i)) + f_{\varepsilon,m}(\sum_{i=1}^s \gamma_i (v_{i,j} + v_{i,j+1}))$$

and

$$\mathrm{hd}_j(\sum_{i=1}^{s}\gamma_i\mathcal{N}_{\varepsilon,m}(v_i)) = \mathrm{hd}_{j-1}(\sum_{i=1}^{s}\gamma_i\mathcal{N}_{\varepsilon,m}(v_i)) + \sum_{i=1}^{s}\gamma_i f_{\varepsilon,m}(v_{i,j}+v_{i,j+1})$$

where $v_i = (v_{i,1},\ldots,v_{i,l})$. By the fact that $f_{\varepsilon,m}$ is a convex function and that $\mathrm{hd}_{j-1}(\mathcal{N}_{\varepsilon,m}(\sum_{i=1}^{s}\gamma_i v_i)) = \mathrm{hd}_{j-1}(\sum_{i=1}^{s}\gamma_i\mathcal{N}_{\varepsilon,m}(v_i))$ we have

$$\mathrm{hd}_j(\mathcal{N}_{\varepsilon,m}(\sum_{i=1}^{s}\gamma_i v_i)) \le \mathrm{hd}_j(\sum_{i=1}^{s}\gamma_i\mathcal{N}_{\varepsilon,m}(v_i)).$$

(The proof for $\mathcal{P}_{\varepsilon,m}$ is analogous). □

Let t' be the minimal t such that b_t is sorted. We assume that $0 < k < l$, and hence $t' > 0$. For $0 \le t \le t'$ let $e_t = mb_t$. For $t > t'$ let $e_t = e_{t'-(t-t') \bmod 2}$.

Claim A.3 *If $t \ge 0$ is even, then*

$$\mathcal{N}_{\varepsilon,m}(e_t) = \mathcal{N}_{\varepsilon,m}(e_{t+1}) = \varepsilon e_t + (1-\varepsilon)e_{t+1}.$$

If $t \ge 1$ is odd, then

$$\mathcal{P}_{\varepsilon,m}(e_t) = \mathcal{P}_{\varepsilon,m}(e_{t+1}) = \varepsilon e_t + (1-\varepsilon)e_{t+1}.$$

Moreover $e_0 \preceq \mathcal{P}_{\varepsilon,m}(e_0)$.

Proof. For each $t \ge 0$ let $e_t = (e_{t,1},\ldots,e_{t,l})$. Let $t+1$ be odd. To see that $\mathcal{N}_{\varepsilon,m}(e_t) = \mathcal{N}_{\varepsilon,m}(e_{t+1})$ note that for odd $i < l$, $e_{t,i} + e_{t,i+1} = e_{t+1,i} + e_{t+1,i+1}$. For each odd $i < l$, $x_i = e_{t,i} + e_{t,i+1} = e_{t+1,i} + e_{t+1,i+1} \in \{0, m, 2m\}$. Let $v = (v_1,\ldots,v_l) = \mathcal{N}_{\varepsilon,m}(e_t)$. Then $v_i = f_{\varepsilon,m}(x_i)$ and $v_{i+1} = g_{\varepsilon,m}(x_i)$. If $x_i = 0$ or $x_i = 2m$, then

$$v_i = x_i/2 = \varepsilon e_{t,i} + (1-\varepsilon)e_{t+1,i}$$

and

$$v_{i+1} = x_i/2 = \varepsilon e_{t,i+1} + (1-\varepsilon)e_{t+1,i+1}.$$

If $x_i = m$, then $e_{t,i} = m$ and $e_{t,i+1} = 0$ (also for $t \ge t'$) and $e_{t+1,i} = 0$ and $e_{t+1,i+1} = m$, hence

$$v_i = \varepsilon x_i = \varepsilon e_{t,i} + (1-\varepsilon)e_{t+1,i}$$

and

$$v_{i+1} = (1-\varepsilon)x_i = \varepsilon e_{t,i+1} + (1-\varepsilon)e_{t+1,i+1}.$$

If l is odd, then

$$v_l = e_{t,l} = e_{t+1,l} = \varepsilon e_{t,l} + (1-\varepsilon)e_{t+1,l}.$$

Thus for all i, $1 \le i \le l$, we have $v_i = \varepsilon e_{t,i} + (1-\varepsilon)e_{t+1,i}$. (The proof for $\mathcal{P}_{\varepsilon,m}$ is analogous.)

The $e_0 \preceq \mathcal{P}_{\varepsilon,m}(e_0)$ follows from the fact that e_0 is the least vector from $[0,m]^l$ in the relation $\preceq$ with the sum of coordinates equal km. □

Corollary A.4 *Let $s > 0$. Let $\gamma_1, \ldots, \gamma_s \geq 0$, such that $\sum_{i=1}^{s} \gamma_i = 1$. Let $\gamma_{s+1} = 0$. Let $\sum_{i=0}^{s} \gamma_i e_i \preceq v$. Then*

$$\sum_{i=0}^{\lfloor s/2 \rfloor} (\varepsilon(\gamma_{2i} + \gamma_{2i+1})e_{2i} + (1-\varepsilon)(\gamma_{2i} + \gamma_{2i+1})e_{2i+1}) \preceq \mathcal{N}_{\varepsilon,m}(v) \tag{6}$$

and

$$\gamma_0 e_0 + \sum_{i=1}^{\lceil s/2 \rceil} (\varepsilon(\gamma_{2i-1} + \gamma_{2i})e_{2i} + (1-\varepsilon)(\gamma_{2i-1} + \gamma_{2i})e_{2i+1}) \preceq \mathcal{P}_{\varepsilon,m}(v). \tag{7}$$

Proof. Equation 6 follows from the fact that by the Claims A.3, A.2 and by Lemma 3.17

$$\sum_{i=0}^{\lfloor s/2 \rfloor} (\varepsilon(\gamma_{2i} + \gamma_{2i+1})e_{2i} + (1-\varepsilon)(\gamma_{2i} + \gamma_{2i+1})e_{2i+1}) = \sum_{i=0}^{s} \gamma_i \mathcal{N}_{\varepsilon,m}(e_i)$$

$$\preceq \mathcal{N}_{\varepsilon,m}(\sum_{i=0}^{s} \gamma_i e_i) \preceq \mathcal{N}_{\varepsilon,m}(v).$$

Analogously we can prove the equation 7. □

Definition A.5 *For $t \geq 0$ and $i \geq 0$ we define the coefficients $\alpha_{t,i}$ as follows:*

- $\alpha_{0,0} = 1$ *and for* $i \geq 1$, $\alpha_{0,i} = 0$.
- *if t is odd, $t \geq 1$, then*

$$\alpha_{t,i} = \begin{cases} \varepsilon(\alpha_{t-1,i} + \alpha_{t-1,i+1}) & \text{if } i \text{ is even, } i \geq 0 \\ (1-\varepsilon)(\alpha_{t-1,i-1} + \alpha_{t-1,i}) & \text{if } i \text{ is odd, } i \geq 1 \end{cases}$$

- *if t is even, $t \geq 2$, then*

$$\alpha_{t,i} = \begin{cases} \alpha_{t-1,0} & \text{if } i = 0 \\ \varepsilon(\alpha_{t-1,i} + \alpha_{t-1,i+1}) & \text{if } i \text{ is odd, } i \geq 1 \\ (1-\varepsilon)(\alpha_{t-1,i-1} + \alpha_{t-1,i}) & \text{if } i \text{ is even, } i \geq 2 \end{cases}$$

Note that, for each $t \geq 0$, $\sum \alpha_{t,j} = 1$ and hence $c_t = \sum_{\alpha_{t,j}>0} \alpha_{t,j} e_j$ is a convex combination of $e_0, \ldots, e_l$. By the Corollary A.4 it is easy to show by induction that:

Claim A.6 *For each $t \geq 0$,*

$$c_t \preceq x_t.$$

Definition A.7 *For $t \geq 1$, $i \geq 0$, let $f_{t,i}$ (the* flow *from i to $i+1$ in step t) be defined as follows:*

$$f_{t,i} = \begin{cases} \alpha_{t-1,i} - \alpha_{t,i} & \text{if } (t \bmod 2) \neq (i \bmod 2) \\ 0 & \text{otherwise} \end{cases}$$

Claim A.8 *Let $t \geq 1$. For all $i \geq 0$, $f_{t,i} \geq 0$ and*

- *If $1 < i < l-1$, then $f_{t+1,i} = (1-\varepsilon)f_{t,i-1} + \varepsilon f_{t,i+1}$*
- *$f_{t+1,1} = \varepsilon f_{t,2}$*

Proof. The first equality can be shown as follows. If $f_{t+1,i} = 0$, then $f_{t,i-1} = 0$ and $f_{t,i+1} = 0$ and the equation follows. If $f_{t+1,i} \neq 0$, then $f_{t+1,i} = \alpha_{t,i} - \alpha_{t+1,i} = \alpha_{t-1,i} + f_{t,i-1} - \varepsilon(\alpha_{t-1,i} + f_{t,i-1} + \alpha_{t-1,i+1} - f_{t,i+1}) = (1-\varepsilon)\alpha_{t-1,i} - \varepsilon\alpha_{t-1,i+1} + (1-\varepsilon)f_{t,i-1} + \varepsilon f_{t,i+1}$. By the Definition A.5 we have $(1-\varepsilon)\alpha_{t-1,i} = \varepsilon\alpha_{t-1,i+1}$ and hence $f_{t+1,i} = (1-\varepsilon)f_{t,i-1} + \varepsilon f_{t,i+1}$. The second equality can be shown in a similar way. □

Definition A.9 *For $t \geq 1$, for any integer i, let $u_{t,i}$ (*upper bound on $f_{t,i}$*) be defined as follows:*

- *$u_{1,0} = 1$, and for $i \neq 0$, $u_{1,i} = 0$, and*
- *for $t > 1$, $u_{t,i} = (1-\varepsilon)u_{t-1,i-1} + \varepsilon u_{t-1,i+1}$*

It is easy to verify the following claim.

Claim A.10 *For $t \geq 1$, $i \geq 0$, $f_{t,i} \leq u_{t,i}$ and for $t \geq 1$, $-t+1 \leq i \leq t-1$, such that $(t \bmod 2) \neq (i \bmod 2)$,*

$$u_{t,i} = \binom{t-1}{(t-1+i)/2} \varepsilon^{(t-1-i)/2}(1-\varepsilon)^{(t-1+i)/2}.$$

□

If $t \geq 1$, and $-t+1 \leq i$, and $i+2 \leq t-1$, then

$$\frac{u_{t,i}}{u_{t,i+2}} = \frac{\binom{t-1}{(t-1+i)/2}}{\binom{t-1}{(t-1+i)/2+1}} \cdot \frac{\varepsilon}{1-\varepsilon} = \frac{t+(i+1)}{t-(i+1)} \cdot \frac{\varepsilon}{1-\varepsilon}$$

For $d > 1$ if $t \geq dl$ and $i < l$, then

$$\frac{u_{t,i}}{u_{t,i+2}} \leq \frac{dl+l}{dl-l} \cdot \frac{\varepsilon}{1-\varepsilon} = \frac{d+1}{d-1} \cdot \frac{\varepsilon}{1-\varepsilon}$$

If $d > 1/(1-2\varepsilon) > 1$, then $c = \frac{d+1}{d-1} \cdot \frac{\varepsilon}{1-\varepsilon} < 1$. Thus $f_{t,i} \leq u_{t,i} \leq c^{(l-i)/2}$. (Indeed, $f_{t,i}$ is either 0 or it is not greater than $u_{t,i} \leq cu_{t,i+2}$ and $u_{t,l} \leq 1$ and $u_{t,l+1} \leq 1$.) On the other hand, if $f_{t,i} > 0$, then

$$f_{t,i} = \alpha_{t-1,i} - \alpha_{t,i} = \alpha_{t-1,i} - \varepsilon(\alpha_{t-1,i} + \alpha_{t-1,i+1})$$

$$\geq (1-\varepsilon)\alpha_{t-1,i} = (1-\varepsilon)^2(\alpha_{t-2,i-1} + \alpha_{t-2,i}).$$

Hence $\alpha_{t-2,i-1} + \alpha_{t-2,i} \leq \frac{1}{(1-\varepsilon)^2}c^{(l-i)/2}$. Thus, for even $t \geq dl+2$ we can estimate the sum of the coefficients $\alpha_{t-2,i}$ with $i \leq l-r$ as follows:

$$\sum_{i=0}^{l-r} \alpha_{t-2,i} = \sum_{0 \leq i \leq l-r,\ f_{t,i}>0} (\alpha_{t-2,i-1} + \alpha_{t-2,i}) \leq \frac{1}{(1-\varepsilon)^2} \sum_{0 \leq i \leq l-r,\ f_{t,i}>0} \sqrt{c}^{(l-i)}$$

$$\leq \frac{1}{(1-\varepsilon)^2} \sum_{i \leq l-r} \sqrt{c}^{(l-i)} \leq \frac{1}{(1-\varepsilon)^2} \cdot \frac{\sqrt{c}^r}{1-\sqrt{c}}$$

We want to find r such, that $\sum_{i=0}^{l-r} \alpha_{t-2,i} < \frac{1}{ml}$. Let

$$r > \frac{2}{\log(1/c)} \left(\log(ml) + \log\left(\frac{1}{(1-\varepsilon)^2(1-\sqrt{c})} \right) \right).$$

Then $(1/\sqrt{c})^r > \frac{ml}{(1-\varepsilon)^2(1-\sqrt{c})}$ and hence

$$\frac{1}{(1-\varepsilon)^2} \cdot \frac{\sqrt{c}^r}{1-\sqrt{c}} < \frac{1}{ml}.$$

If

$$ml \geq \frac{1}{(1-\varepsilon)^2(1-\sqrt{c})},$$

then we can have any r such that

$$r > \frac{4}{\log(1/c)} \log(ml).$$

Note that for $d = 4/(1-2\varepsilon)$,

$$c = \frac{5-2\varepsilon}{3+2\varepsilon} \cdot \frac{\varepsilon}{1-\varepsilon}$$

and, for $0 < \varepsilon < \frac{1}{3}$,

$$c < \frac{13}{22}$$

and hence

$$\frac{1}{(1-\varepsilon)^2(1-\sqrt{c})} < \frac{9}{4\left(1-\sqrt{\frac{13}{22}}\right)} < 12.$$

Recall that

$$c_{t-2} = \sum_{\alpha_{t-2,i}>0} \alpha_{t-2,i} e_i.$$

Let $c_{t-2,j}$ be the j-th coordinate of c_{t-2}, where $j < l-k-r$. By Claim A.1, e_i has only zeroes as the j-th coordinate if $i > l-r$. Thus

$$c_{t-2,j} \leq m \sum_{i=0}^{l-r} \alpha_{t-2,i} < 1/l.$$

Hence $\mathrm{hd}_{l-r}(c_{t-2}) < (l-r)/l < 1$. Recall that $c_{t-2} \preceq x_{t-2}$. Thus we have $\mathrm{hd}_{l-r}(x_{t-2}) < 1$.

Lemma 3.21 follows if we take $\alpha = d \geq 4/(1-2\varepsilon)$ and $\beta \geq \frac{4}{\log(1/c)}$.